NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

OBSTACLE DETECTION AND AVOIDANCE ON A MOBILE ROBOTIC PLATFORM USING ACTIVE DEPTH SENSING

by

Taylor K. Calibo

June 2014

Thesis Advisor:	Xiaoping Yun
Second Reader:	Zac Staples

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2014	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE OBSTACLE DETECTION AND AVOIDANCE ON A MOBILE ROBOTIC PLATFORM USING ACTIVE DEPTH SENSING			**5. FUNDING NUMBERS**
6. AUTHOR(S) Taylor K. Calibo			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

The ability to recognize and navigate surrounding environments free from collision with obstacles has been at the forefront of mobile robotic applications since its inception. At the price of nearly one tenth of a laser range finder, the Xbox Kinect uses an infrared projector and camera to capture images of its environment in three dimensions. The objective of this thesis was to investigate if the Xbox Kinect can be utilized to detect thin or narrow obstacles that are often invisible to the P3-DX mobile robotic platform. We present an algorithm to process and analyze point cloud data from the Xbox Kinect sensor and transform it into a two-dimensional map of the surrounding environment for further use with the P3-DX. Obstacle avoidance scenarios were then performed using two separate algorithms: a narrow corridor following algorithm and a potential fields algorithm. The results demonstrate that in a structured testing environment, the Xbox Kinect can be used to detect and avoid narrow obstacles that are not immediately recognized by the onboard sonar array of the P3-DX.

14. SUBJECT TERMS Mobile Robotics, Xbox Kinect, Obstacle Avoidance, P3-DX, Potential Fields, MATLAB			**15. NUMBER OF PAGES** 75
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

OBSTACLE DETECTION AND AVOIDANCE ON A MOBILE ROBOTIC PLATFORM USING ACTIVE DEPTH SENSING

Taylor K. Calibo
Ensign, United States Navy
B.S., United States Naval Academy, 2013

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2014

Author:	Taylor K. Calibo
Approved by:	Xiaoping Yun Thesis Advisor
	Zac Staples Second Reader
	R. Clark Robertson Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The ability to recognize and navigate surrounding environments free from collision with obstacles has been at the forefront of mobile robotic applications since its inception. At the price of nearly one tenth of a laser range finder, the Xbox Kinect uses an infrared projector and camera to capture images of its environment in three dimensions. The objective of this thesis was to investigate if the Xbox Kinect can be utilized to detect thin or narrow obstacles that are often invisible to the P3-DX mobile robotic platform. We present an algorithm to process and analyze point cloud data from the Xbox Kinect sensor and transform it into a two-dimensional map of the surrounding environment for further use with the P3-DX. Obstacle avoidance scenarios were then performed using two separate algorithms: a narrow corridor following algorithm and a potential fields algorithm. The results demonstrate that in a structured testing environment, the Xbox Kinect can be used to detect and avoid narrow obstacles that are not immediately recognized by the onboard sonar array of the P3-DX.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

EXECUTIVE SUMMARY

The ability to recognize and navigate surrounding environments free from collision with obstacles has been at the forefront of mobile robotic applications since its inception. As today's regional conflicts are likely to occur in populated urban areas where the terrain is often unpredictable and the threat to non-combatant life is high, a recent trend in mobile robotics has been toward small, low-cost platforms to provide reconnaissance information [1]. For any mobile robotic platform, obstacle avoidance is a key qualification to ensure the safety of the human user and overall function of the platform itself. [2]

Previous research in obstacle avoidance has focused on using systems that use large cameras such as computer vision systems or laser rangefinders that cost up to several orders of magnitude more than the inexpensive autonomous systems on which it would be deployed [3], [4]. The Xbox Kinect, a product created by the Microsoft Corporation, is a low-cost IR sensor that was originally used for gaming applications. With its many uses that range from distinguishing different types of objects to returning depth information of a human operator, the Kinect can act as an improvement over existing sonar methods with its unique capability to capture three-dimensional images of its surrounding environment. The P3-DX is a small (9 kg) two-wheel research and development robot that uses an active sonar ranging system to detect and avoid obstacles in indoor environments. With 16 separate sonars, the robot is able to run obstacle avoidance algorithms in rooms with large obstacles; however, its sonar system lacks the angular resolution to identify smaller objects in its path. In order to address the need for a low-cost imaging device that can be utilized for obstacle avoidance, the Xbox Kinect is investigated as a reliable improvement to the current sonar system onboard the P3-DX in this thesis. We present an algorithm to analyze depth data from the Xbox Kinect sensor and transform it into a two-dimensional (2-D) map of the surrounding environment.

The specific characteristics and limitations of the Kinect were compared with the sonar array onboard the P3-DX platform, and an algorithm was produced that segmented the three-dimensional depth images from the Kinect into 57 different ranges, each

representing one angle in the sensors field-of-view. These ranges were transformed from the sensors coordinate system to the robot coordinate system in order to provide a higher resolution map in the field of view of the sensor. An example of the Kinect capability to map narrow objects is demonstrated in Figure 1.

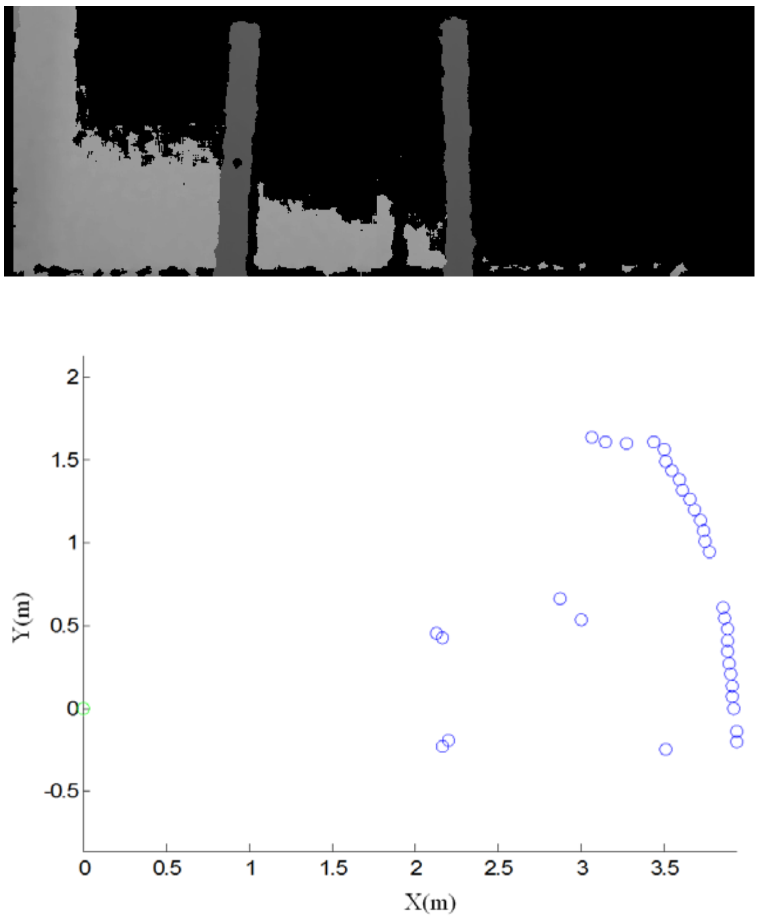


Figure 1. The depth image, top, of two thin poles captured by the Xbox Kinect is shown. The returns (blue) from the top image are transformed to provide a 2-D map of the environment.

At the top of Figure 1, a depth image from the Kinect displays two poles represented as dark grey narrow columns with a lighter background behind them. The corresponding two-dimensional map output by the mapping algorithm is shown on the bottom of Figure 1. Two clusters of data points can be identified at approximately two meters from the origin where the robot (green) is located. Other returns shown in the map with the *x*-axis coordinate greater than 2.5 are from background objects. The map corresponding to Figure 1 for the sonar array (not shown) on the P3-DX does not present any forward facing returns.

Several more objects were mapped into the two-dimensional plane and compared with the returns provided by the sonar array previously used by the P3-DX robot. The results are encouraging because the resolution provided by the Kinect is much greater than what the sonar system was able to provide.

Obstacle avoidance scenarios were then performed using two separate algorithms, a narrow corridor following algorithm and a potential fields algorithm. The results demonstrated that in a structured testing environment, the Xbox Kinect can be used to detect and avoid narrow obstacles that are not immediately recognized by the onboard sonar array. A proof-of-concept design that demonstrates the Xbox Kinect can be used to reduce error and improve accuracy in indoor environments for robot navigation was provided by this research.

LIST OF REFERENCES

[1] J. Pransky, "Mobile robots: Big benefits for U.S. Military," *Industrial Robot: An International Journal*, vol. 24, no. 2, pp.126–130, Jan. 1997.

[2] R. Simpson, D. Poirot, and F. Baxter, "The Hephaestus Smart Wheelchair System," *IEEE Transactions on, Neural Systems and Rehabilitation Engineering*, vol. 10, no. 2, pp. 118–122, June 2002.

[3] C. Ye, "Mixed pixels removal of a Laser Rangefinder for mobile robot 3-D terrain mapping," *International Conference on Information and Automation, ICIA 2008*, pp. 1153–1158, June 2008.

[4] B. Wei, J. Gao, K. Li, Y. Fan, X. Gao and B. Gao, "Indoor mobile robot obstacle detection based on linear structured light vision system," *in IEEE International Conference on Robotics and Biomimetics*, Feb. 2009, pp. 834–839.

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGEMENTS

I would like to thank Professor Xiaoping Yun for being my advisor and CDR Staples for being my constant source of motivation in the field of mobile robotics. I would also like to thank James Calusdian for the hours he spent with me in the lab, ensuring that I had everything that I needed to be successful in my research. Additionally, I would like to thank Capt Joshua Lum and LT Dirk Lundgren who gave me assistance in operating the P3-DX and help with coding. Additionally, I would like to thank LT Brian Schaus and LT Thomas Manemeit for their help and guidance in areas outside of my thesis research. I would also like to thank the Naval Academy and the Bowman Scholarship program for allowing me the opportunity to complete my graduate education at Naval Postgraduate School. Lastly, I would like to give a special thanks to my family, Mark Calibo, Tandy Calibo and Michael Calibo as well as my friends William McGee, Kristijonas Fussman and Gabe Garcia for their unconditional support in everything that I have chosen to do in my life.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The ability to realize and navigate surrounding environments free from collision with obstacles has been at the forefront of mobile robotic applications since its inception. As today’s regional conflicts are likely to occur in populated urban areas where the terrain is often unpredictable and the threat to non-combatant life is high, a recent trend in mobile robotics has been toward small, low-cost platforms to provide reconnaissance information [1]. Recent research has attempted to improve upon obstacle avoidance methods by investigating various types of sensors ranging from sonars to laser scanners. The limiting factor for sensor deployment on small inexpensive robotic platforms has been in both power and cost of these sensors.

On November 4, 2010, Microsoft Corporation released the Xbox Kinect, a low-cost controller-free sensor that uses a color camera (RGB) and three-dimensional (3-D) depth ranging sensor designed primarily for gaming applications. This design caught the eye of developers and engineers alike and reverse engineering efforts led to the creation of open source drivers for use in the mobile robotics community.

At the price of nearly one tenth of a laser range finder, the Kinect provides a cost effective solution using infrared light to generate voxels (depth pixels), consequently adding a third dimension to the image received by the sensor. In combination with the RGB camera, the data provided by the Kinect has proven to be very useful in low-cost designs requiring visual identification of its surrounding environment.

Previous research has shown the Xbox Kinect to be an invaluable device in modeling indoor environments for robot navigation. In addition to the environment, the Kinect has been shown to have significant success in separating a human object from his/her surroundings [2]. The system still has many limitations, which include limited detection range from about 50 cm to 6 m [3]. Additionally, the projection pattern in very bright or low light settings is generally absorbed by the environment and not seen by the IR camera. With these considerations, best practices in previous research have shown the optimal environment for using the capabilities of the Kinect for obstacle avoidance and human object detection is indoors.

The purpose of this research is to provide a proof-of-concept utilizing research to develop and implement an obstacle detection and avoidance algorithm on the Pioneer P3-DX platform using the Xbox Kinect and MATLAB software. The primary phase demonstrates the Kinect’s environment modeling capability, which should greatly reduce error and improve accuracy over its current sonar sensor array. The secondary phase processes the visual data captured by the Kinect, implements it in a control scenario, and provides obstacle avoidance capabilities. The purpose of the secondary phase is focused on expanding the practical benefits of the Xbox Kinect sensor such as using the mobile platform to track and follow human objects in future work.

This thesis is divided into seven chapters. A literature review of the pertinent research on the Xbox Kinect and its use in mobile robotics is provided in Chapter II. The characteristics of the Kinect, including its strengths and weaknesses as well as the mobile robotic platform on which it is mounted, is the focus of Chapter III. The environment modeling method developed in software to support translating three-dimensional images into a two- dimensional coordinate space are discussed in Chapter IV. The path planning algorithm used to direct the mobile robotic platform are discussed in Chapter V. The results of the experimentation and effectiveness of the algorithm designed are discussed in Chapter VI. A conclusion and a discussion of future work are provided in Chapter VII.

II. BACKGROUND

From dexterous humanoids that provide the capability to build machines in space [4] or smart wheelchairs that give handicapped users more independence [5], new mobile robotic applications arise each day that are designed with mobility as their primary purpose. For any mobile robot, obstacle avoidance is a key qualification to ensure the safety of the human user and overall function of the platform itself.

One particular area of important consideration for mass deployment on mobile robots is in the sensor systems used to identify obstacles and map the surrounding environment. Current systems have shown several weaknesses in the form of cost, type of data generated and power consumption of their sensors. With the 2009 release of the Xbox Kinect, various research studies have been conducted to determine the potential of the Kinect to provide improvements to the aforementioned problems of current sensor systems. Previous work done in the field of mobile robotics with an emphasis on the areas that can be improved through use of active 3-D depth sensing with the Xbox Kinect is discussed in this chapter.

There has been a significant amount of research done using the sonar with the P3-DX pioneer mobile robot to gather information about the robot's surroundings as well as map its environment [6], [7]. The primary function of the P3-DX sonar array is to provide a baseline for detection of the surrounding environment at a very low-cost; however, the data that is provided by this sonar system is less reliable and does not address the needs of environments that have thin pole-like objects, such as the stem structure of an office chair or the legs of the table. The Xbox Kinect can act as an improvement over currently existing sonar methods due to its unique capability to capture three-dimensional images of the surrounding environment. This provides the ability to identify objects that are not as easily identifiable using minimal sonar returns.

As will be discussed in Chapter III, the resolution of the P3-DX sonar systems is smaller and less reliable than ultrasonic [8] or laser range-finding systems [9] used to detect large obstacles in more complex environments. The laser rangefinder can provide much higher resolution images to the data processing algorithm; however, systems such as the Sick LIDAR range-finding system are costly and large. This makes them less deployable on non-ground based autonomous systems. Additionally, these systems cannot see small objects that may be resting on the surface of another object such as a floor or a table top. The system itself only provides a “slice” of the robotic environment and is less useful in identifying useful features of the surrounding environment.

Additionally, research on path planning using the P3-DX has been done using large wireless cameras that provides the desired image of an unknown environment [10]. Image planning techniques provide a larger depth-of-field based on line-of-sight of the camera and can often provide a nearly optimal path based on one image. These techniques, however, require a tremendous amount of processing power and large cameras that are unable to be deployed on smaller autonomous systems. Other vision processing systems [11] use methods that involve the projection of linear structured light to identify irregularities in objects located in front of the camera. The Xbox Kinect uses a similar method known as light coding to identify distortion or irregularities in obstacles; however, it does it at a much cheaper cost than the previously mentioned light sensors. The characteristics of this infrared (IR) sensor used on the Xbox Kinect are further discussed in Chapter III.

III. HARDWARE AND SOFTWARE DESCRIPTIONS

A. HARDWARE

The Kinect color camera provides RGB video at a frame rate of 30 Hz and outputs 8-bit VGA resolution (640×480 pixels) with a Bayer color filter. The depth sensor is separated from the color camera by 7.5 cm and also provides a VGA stream (640×480 pixels) using an 11-bit depth that can operate with 2048 levels of sensitivity [12]. The device is shown in Figure 1 with the annotated locations of the IR projector, RGB camera and IR camera.

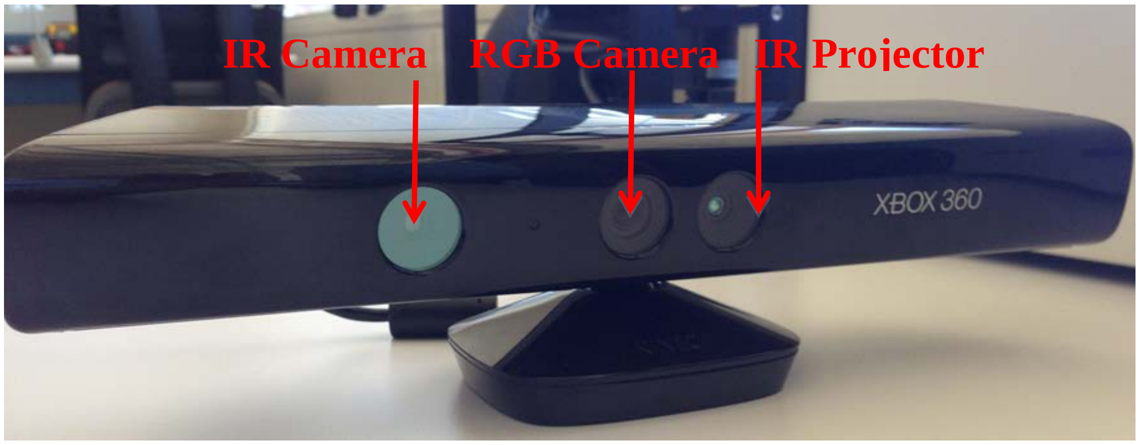


Figure 1. The Xbox Kinect is shown along with the locations of its three sensors.

Although the exact method in which the Kinect provides three-dimensional depth sensing has not yet been published by its manufacturer, PrimeSense, reverse engineering efforts have led to a better understanding of how the device returns a three-dimensional point cloud to the user. The IR sensor projects a memory-saved fixed pseudorandom pattern which is collected by the IR camera in a process known as stereo triangulation [13]. An image of the process described above is shown in Figure 2.

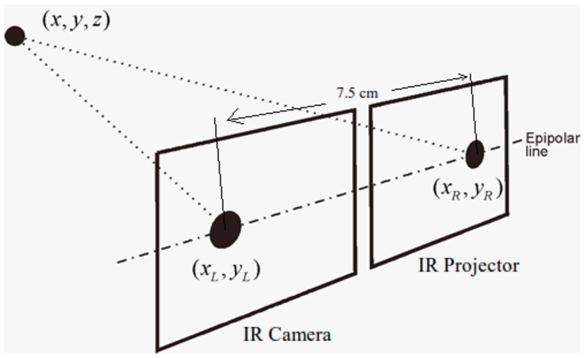


Figure 2. Method of stereo triangulation used to record depth on the Kinect, from [13].

Through the use of this method of light coding, the Kinect is able to return an array of voxels that represent objects at a distance in meters appearing in front of the camera. Unfortunately, due to its relatively low power and geometry, this method of depth detection is limited in range of practical use. Previous research [14] has shown that the Xbox Kinect is unable to locate objects closer than 0.5 m and subsequently returns a zero for any objects detected in this range. In a similar manner, the infrared camera is limited by the strength of the infrared projector and is unable to detect objects further than six meters. The validity of these statements can be tested in a laboratory setting to ensure that both the calibration of the sensor and the output is correct. The effects of these limitations make it unlikely that the Kinect can be used in applications where longer range returns are required; however, for indoor and confined spaces the Kinect Computer vision proves to be more than adequate.

The sample images shown in Figure 3 are both an image captured by the RGB sensor (left) as well as an image captured by the depth Sensor (right). The image shown on the right of Figure 3 provides a 640×480 depth image where each pixel represents the distance in meters. The pixels in lighter shades of gray represent distances farther away from the sensor, whereas the pixels in darker shades are closer to the sensor. Depths that are completely black represent a non-return from the sensor. In order to better distinguish the non-returns from returns, black pixels were colored red as shown in Figure 4. Some areas behind the chair are areas of non-return due to a parallax effect caused by the distance between the IR projector and IR camera [14]. A larger parallax effect can be seen when the angle of elevation on the Kinect camera is increased.

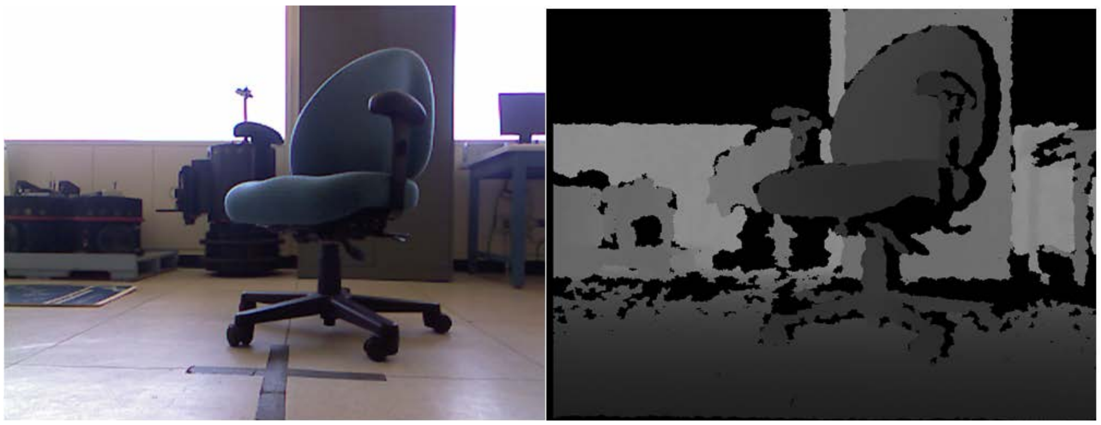


Figure 3. Sample image of desk chair taken with Kinect RGB camera, left, and Kinect depth senor, right are shown. In the depth image, darker pixels are closer to the sensor while lighter pixels are further away. Areas that are completely black do not have returns.

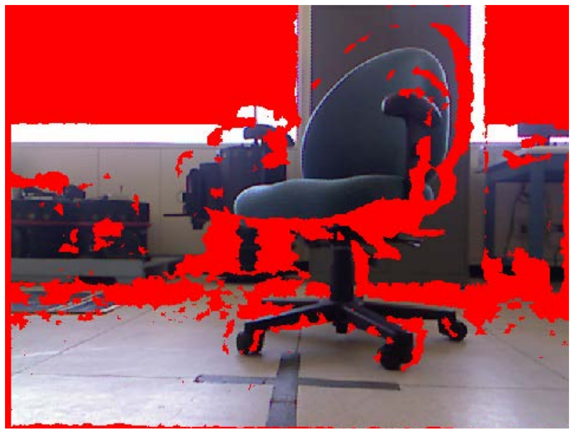


Figure 4. Sample RGB image from the Kinect sensor with non-returns superimposed. Areas in red are points of non-return. Areas of non-returns outlining the chair are due to a parallax effect that exists due to the geometry of the camera.

B. MOBILE ROBOTIC PLATFORM

The robotic platform used for testing the Xbox Kinect for obstacle avoidance was the Pioneer P3-DX. The P3-DX is a small (9 kg) two-wheel research and development robot differentially driven with a single, free rotating caster wheel. The robot is built with a strong aluminum body and has a variety of expansion power ports to allow the robot to interface with server software. The robot features an onboard Hitachi H8s microprocessor for low-level tasks such as sensor data collection, motor control and communication with higher level systems through a serial communications port. The P3-DX also has a sonar array of 16 sonars separated by 15 degree increments which can be used to provide depth readings. A top view diagram of the sonar is shown in Figure 5 on the left. A picture of the P3-DX outfitted with the Xbox Kinect is shown in Figure 5 on the right.

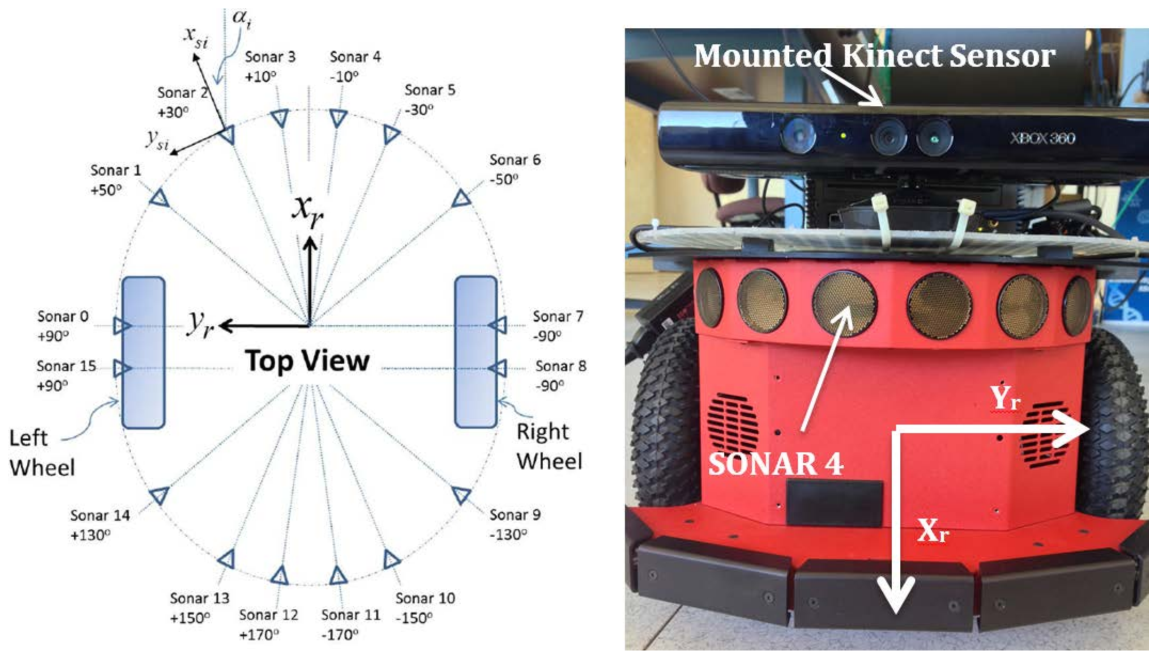


Figure 5. The sonar array system of the P3-DX. The top down view with locations of the sonar array is shown, left, from [15], and the P3-DX with the Kinect mounted, right.

C. SOFTWARE

The P3-DX robot was tethered to a remote desktop through a NetGear N150 wireless receiver to a mini-computer mounted on the top platform of the robot. For this thesis, all algorithms were created and run in MATLAB R2013a specifically for controlling the P3-DX robotic platform. The raw data from the Kinect were imported using the Kinect for Windows Adaptor from Image Acquisition Toolbox [16].

IV. METHODOLOGY

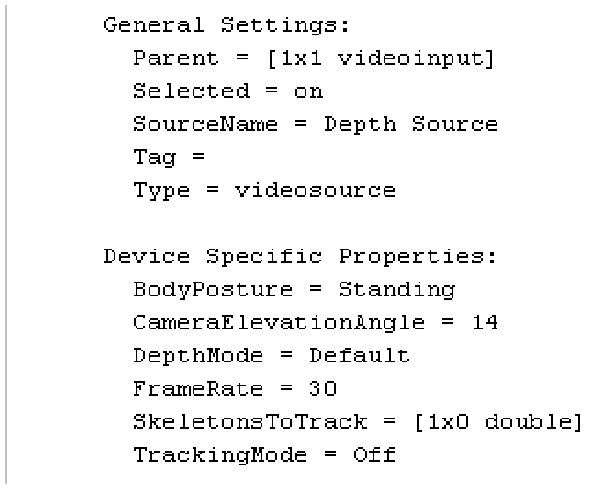
The obstacle detection algorithm involves three main sections. The first is to transform the depth image to a three-dimensional point cloud in MATLAB using the processing hardware on board the Xbox Kinect. The second is to create a two-dimensional (2-D) occupancy map by projecting the points collected by the depth sensor onto a 2-D top-down coordinate map. Finally, the obstacle avoidance algorithm decides how to move the robot based on the robots location relative to the obstacle.

A. PROCESSING THE DEPTH IMAGE

The Kinect for Windows Adaptor found in the Image Acquisition Toolbox [16] was used to acquire images using the Kinect device and MATLAB software. The Kinect outputs four data streams: image, depth, skeletal and audio. The image and depth stream is utilized in this thesis to provide 3-D returns from the sensor to MATLAB. The audio stream was not used.

The image stream contains information returned by the RGB color sensor in various color formats. The depth stream is returned by the depth sensor. The stream contains a 640×480 pixel grid that contains depth information in millimeters at a frame rate of 30 frames per second. The image acquisition toolbox typically recognizes separately installed imaging devices and assigns a device ID to each. Since the Kinect has two sensors, separate Device ID’s were assigned to both the RGB camera and the depth sensor. In order to stream the information from the Kinect to MATLAB, the Kinect was installed on a small microcomputer mounted on the platform of the P3-DX robot and subsequently tethered to the robot itself.

Video input objects were then created for the both the color sensor and the depth sensor. The properties of each device are shown in the display summary shown in Figure 6.



```
General Settings:
  Parent = [1x1 videoinput]
  Selected = on
  SourceName = Depth Source
  Tag =
  Type = videosource

Device Specific Properties:
  BodyPosture = Standing
  CameraElevationAngle = 14
  DepthMode = Default
  FrameRate = 30
  SkeletonsToTrack = [1x0 double]
  TrackingMode = Off
```

Figure 6. Display summary for the video source object created in MATLAB.

In the display summary, CameraElevationAngle controls the pitch of the sensor lens and must be an integer between -27 and 27 degrees. A value of 14 degrees in elevation was selected for the purposes of eliminating artifacts collected by the infrared sensor as they were reflected of the ground.

A comparison of depth images taken in a long hallway for both zero and 14 degrees of elevation are shown from the mounted Kinect sensor as shown in Figure 7. In these images, the ground plane appears in the image on the left. The ground plane makes it appear to the sensor that there is an object in the front direction rather than a path to travel over. When the angle of the camera is tilted upward, the ground plane disappears. The plane is instead replaced by the ceiling as is shown in the image on the right of Figure 7. Rather than creating an algorithm to extract the ground, changing the elevation is preferable for recognizing objects at distances further away than 500 cm and saves processing time to find and remove the extra data points. Although the ceiling plane appears in the image on the right, the image is further cropped to only extract the returns that exist between 280 and 480 pixels. The final cropped image is shown in Figure 8.

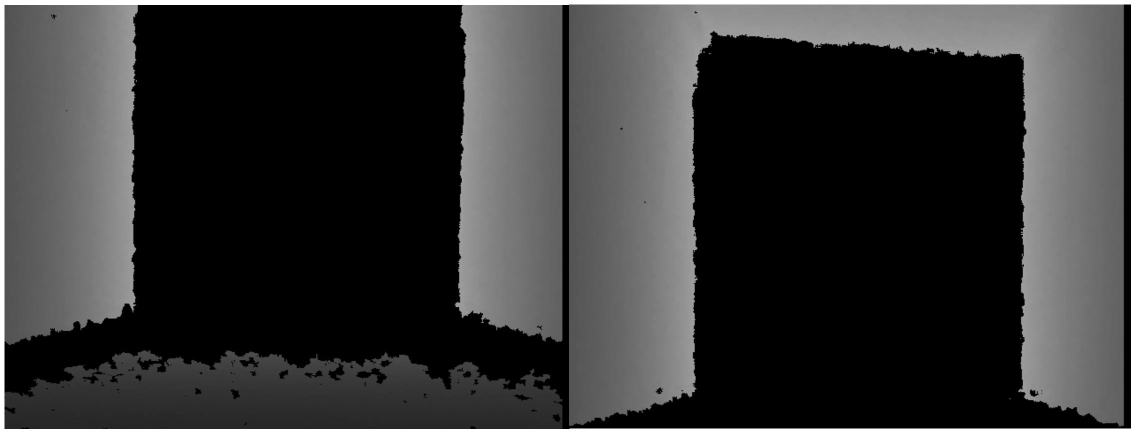


Figure 7. A comparison of depth maps for zero and 14 degrees angle of elevation in a long hallway are shown above. A depth image captured at an angle of elevation of zero degrees, left. A depth image captured at an angle of elevation of 14 degrees, right.

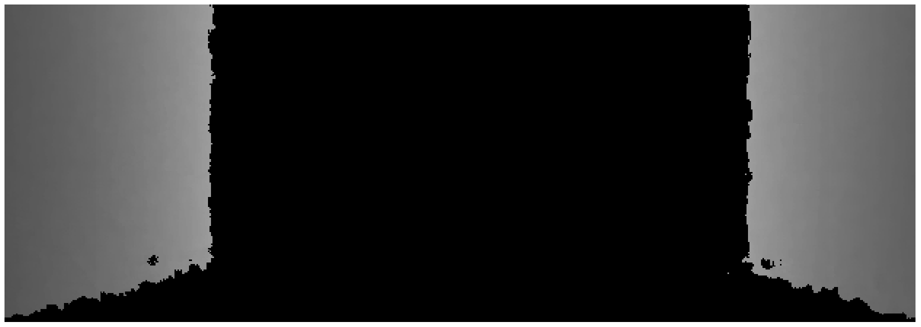


Figure 8. The final depth image after being cropped between 280 and 480 pixels.

The imaging method was applied to a room with obstacles as well. Two wooden planks were set up in an opening in front of the P3-DX outfitted with the Xbox Kinect. An image captured by the RGB camera can be seen in Figure 9, while the corresponding cropped depth map is shown in Figure 10. The ability of the Kinect to display the two thin obstacles shows the capability of the Kinect’s resolution over that of the sonar ranging array since the Kinect provides a quick and accurate range resolution in a complex environment. Both wooden planks are shown with little loss of resolution, and the ground plane is returned as an array of zeros (non-returns), demonstrating the Kinect’s ability to distinguish between flat surface and a raised object.

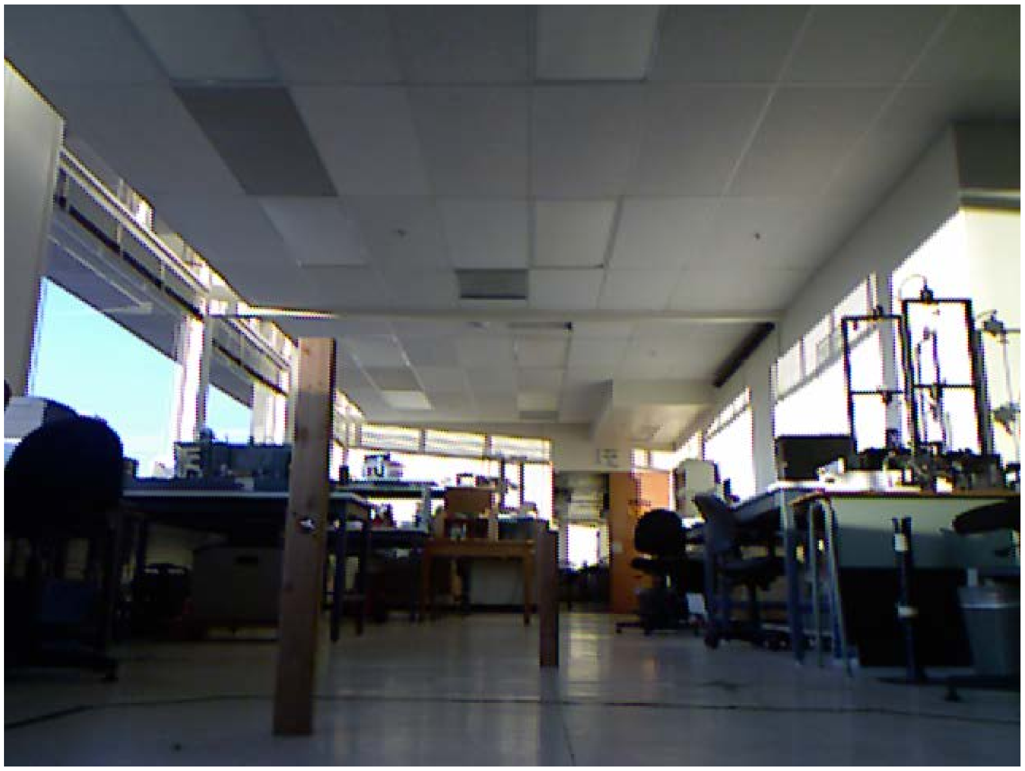


Figure 9. RGB image captured by the camera on the Xbox Kinect.

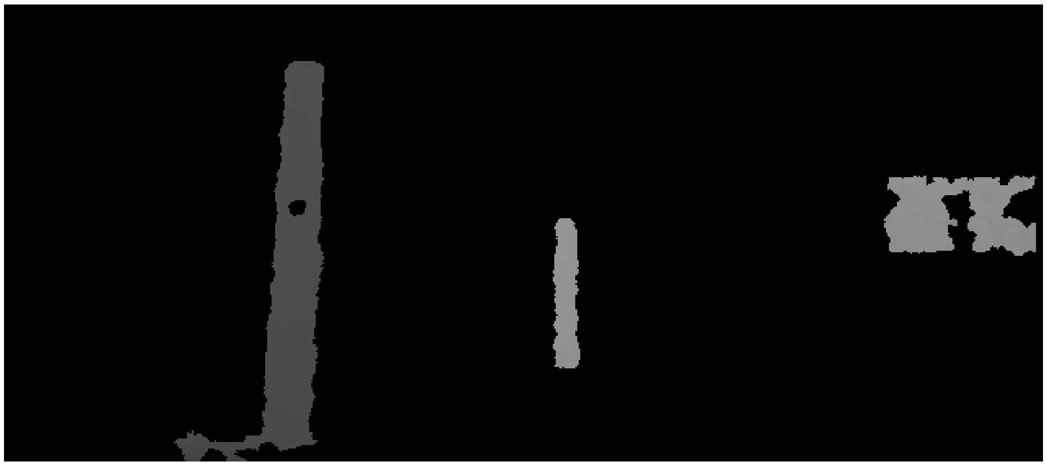


Figure 10. Post-processed depth image output from the Kinect.

In addition to the settings that control the physical characteristics of the camera, another important mode to set in the Kinect is the manual trigger mode. This mode allows the user to monitor the video stream being acquired using the preview function. In addition, it provides a more efficient method of manually executing the trigger while in a loop. The depth mode is set to default and indicates the range of depth in the depth map. The default setting sets the range from 50.0 to 400.0 cm.

B. SENSOR CALIBRATION

One of the primary purposes for conducting this research was to assess the capabilities of the Kinect against the previously used sonar system on board the Pioneer P3-DX robot. The Xbox Kinect user manual provides that the Kinect performs with a depth detection range of 0.8 to 3.5 m along with an angular field-of-view of 57 degrees horizontal by 43 degrees vertical [7]. The Kinect and sonar ranging system were tested on the same robotic platform using a white Styrofoam board and a tape measure in meters. A plot of the observed data in meters using the Kinect and the Sonar systems versus the actual distance using the tape measure is shown in Figure 11.

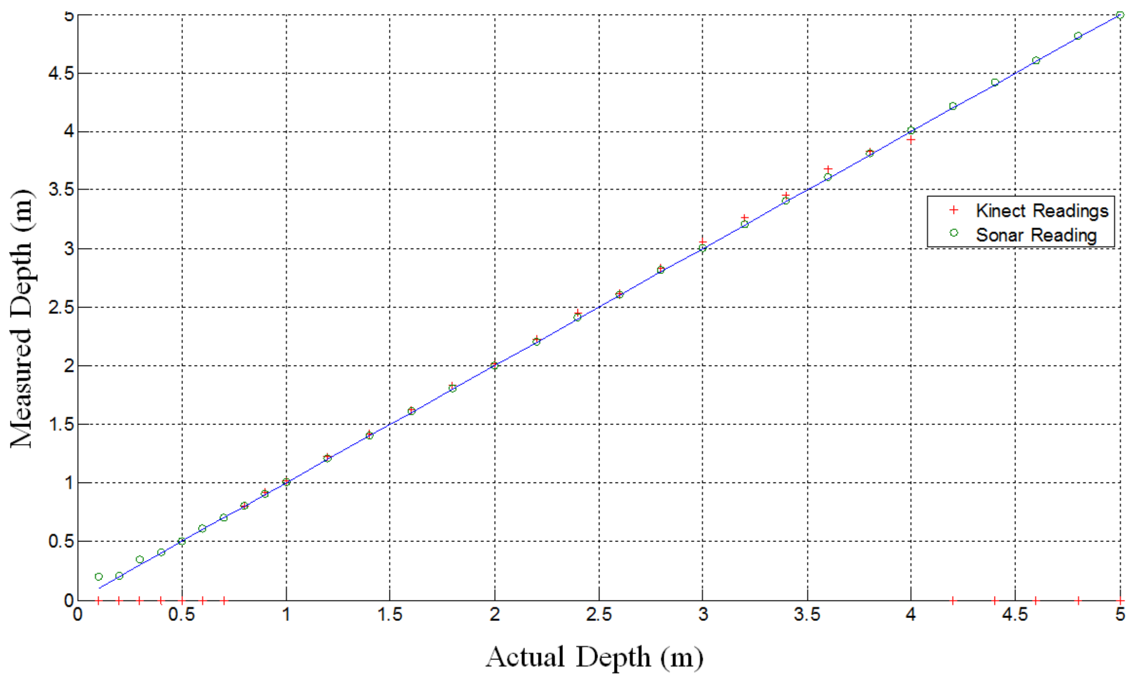


Figure 11. Measured depth vs actual depth for the P3-DX sonar array and the Xbox Kinect sensor.

The measurements were conducted inside a moderately lit computer laboratory and were taken using 10.0 cm increments between 0.0 m and 1.0 m and followed by measurements at 200.0 cm increments for the remaining 4.0 m. As shown in Figure 11, the linear characteristic of both the P3-DX sonar array and the Xbox Kinect demonstrate the capability of both systems to accurately assess depth; however, as is illustrated in Figure 11, the Xbox Kinect only provides returns between 0.8 m and 4.0 m, whereas the sonar system gives accurate depth measurements between 0.2 m and 5.0 m. As will become apparent, the obstacle avoidance algorithm does not require accuracy levels that are less than what is already provided by the sonar system or the Xbox Kinect measurements to accurately navigate obstacles in an indoor environment.

C. 2-D COORDINATE MAPPING

The purpose of this section is to illustrate the method of transforming a depth image into a two-dimensional coordinate plane of distance points. In this study, we used the depth image that was generated using all of the features selected in Section A of this chapter.

In order to accurately map the environment surrounding the robot, artifacts must first be removed from the depth image. Because the robot is close to half a meter in height, the depth image was cropped vertically between 280 and 480 pixels to represent all objects in the environment that lie between the minimum and maximum height of the robot. In addition to omitting objects taller than the robot, the cropping ensured any additional IR returns from the ceiling were also omitted.

Recall from Chapter III, the Xbox Kinect can be treated as a normal camera with an angular field of view of 57 degrees. In order to ensure the best estimate of object distance from the IR sensor, the depth image was horizontally cropped to a length of 627 pixels so that depths could easily be segmented into 57 separate units. Each unit contains 11 horizontal pixels to represent one degree of angular resolution. A diagram of the cropped image fed into the 2-D coordinate transformation algorithm is shown in Figure 12.

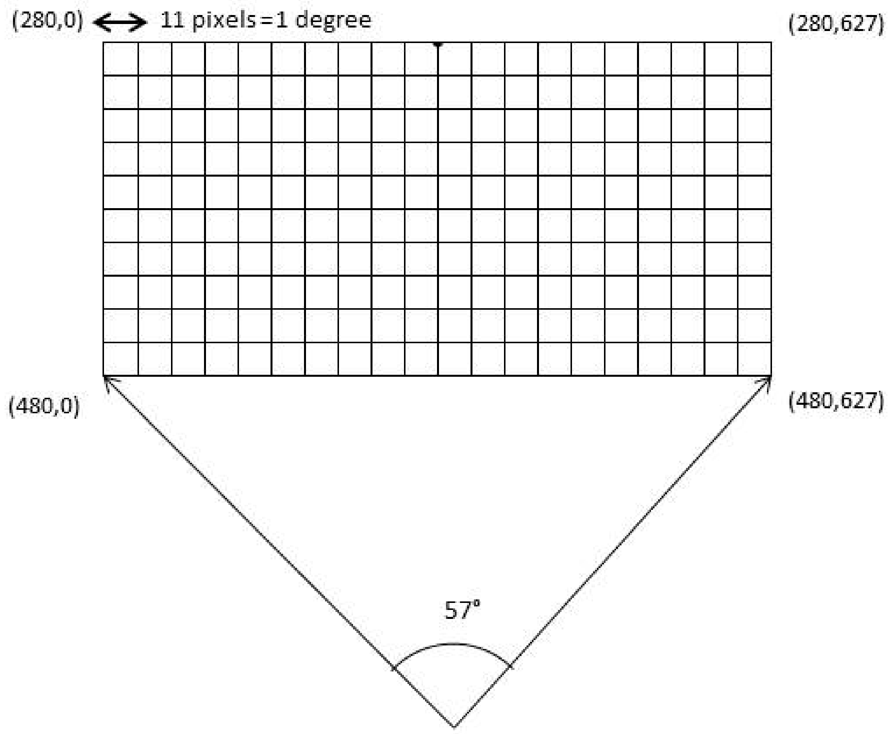


Figure 12. Diagram of cropped image imported into the 2-D coordinate mapping algorithm.

1. Definition of Transformation Matrix

In order to use the information contained in the depth image provided by the Xbox Kinect, we must first be able to represent the positions of objects in the surrounding environment in terms of the robot frame of reference. The top-down diagram of the robotic frame illustrated in Figure 5 for the sonar array places the coordinate system in the center of the robot. Because the IR sensor was mounted to the front of the robot, we defined our origin relative to the location of the Kinect.

The position of an object in the surrounding environment of the robot can be defined using a 3×1 position vector. In this thesis, vectors are written with a leading superscript indicating the coordinate system to which they are referenced. An example of this, where *A* represents the coordinate system to which the position vector belongs with its corresponding components represented in lowercase, is given by

$$^{A}P = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}. \tag{1}$$

The position of an object relative to the robot frame of reference may therefore be defined as the individual elements of the position vector at a certain orientation relative the robot. This transformation can be made using a rotation matrix which defines the principal directions of a coordinate system relative to another. We name it with the notation ${}^{A}_{B}R$, where the rotation describes the coordinate system *B* relative to *A*. The resulting transformation for the position of an object in the robot frame of reference is

$$^{R}P = {}^{R}_{S}R\,{}^{S}P + P_{Sorg} \tag{2}$$

where ^{R}P is the position of the observed object relative to the robot, ${}^{R}_{S}R$ is the rotation matrix with sensor coordinate frame relative to the robot coordinate frame, ^{S}P is the object position relative to the sensor, and P_{Sorg} is the vector that locates the sensors origin in the robot frame of reference.

Since a robot which makes turns only in two dimensions is used for this research, all rotations are about the *z*-axis and use the corresponding rotation matrix given by

$${}^{R}_{S}T = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3}$$

To further simply the calculation shown in Equation (2), we define a matrix operator known as a transformation matrix ${}^{R}_{S}T$ from the sonar coordinate system to the robot coordinate system. This allows us to think of Equation (2) as a mapping from one frame to another in matrix form and aids in writing of a more compact expression, which is

$$^{P}R = {}^{R}_{S}T\,{}^{S}P. \tag{4}$$

The structure of the expression shown in Equation (4) after Equation (3) has been substituted is

$$\begin{bmatrix} {}^{R}P_x \\ {}^{R}P_y \\ {}^{R}P_z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & P_{Sorg\_X} \\ \sin(\theta_i) & \cos(\theta_i) & 0 & P_{Sorg\_y} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{S}P_x \\ {}^{S}P_y \\ 0 \\ 1 \end{bmatrix}. \tag{5}$$

Note that for the purposes of this thesis, any ${}^{R}P_Z$ that is returned will be zero since the robot is only operating within a two-dimensional space. If the robot were to move up a ramp or move out of the two-dimensional plane, the P_{Sorg_z} would take on a value and provide a return in a third dimension.

2. Data Processing Algorithm

Once the cropped image has been imported successfully into MATLAB, the algorithm uses sequential averaging to create 57 data points to represent each angle in the sensors field of view. The algorithm removes all of the non-returns by placing an empty matrix inside all of the pixels with a zero depth-of-field. The 11 pixels are then averaged by column, and the pixel with the depth with the minimum distance to the sensor is selected and stored as the depth for that angle. Once a data point for each angle in the field-of-view has been stored, the loop repeats and creates a new set of data points. For ease of visualization, a diagram of the combined coordinate systems along with the Kinect sensor's field-of-view is shown in Figure 13. In the Figure, S_n relates to the segmented image which treats each of the 57 segments as if they were individual sensors. Points were arbitrarily selected on the fictitious obstacles to demonstrate the position of the *x* and *y* components for each sensor. The red vectors labeled ${}^{R}P_x$ and ${}^{R}P_y$ show the mapping of the position vector in the S_n in the robot coordinate system where they can be further used in the object avoidance algorithm.

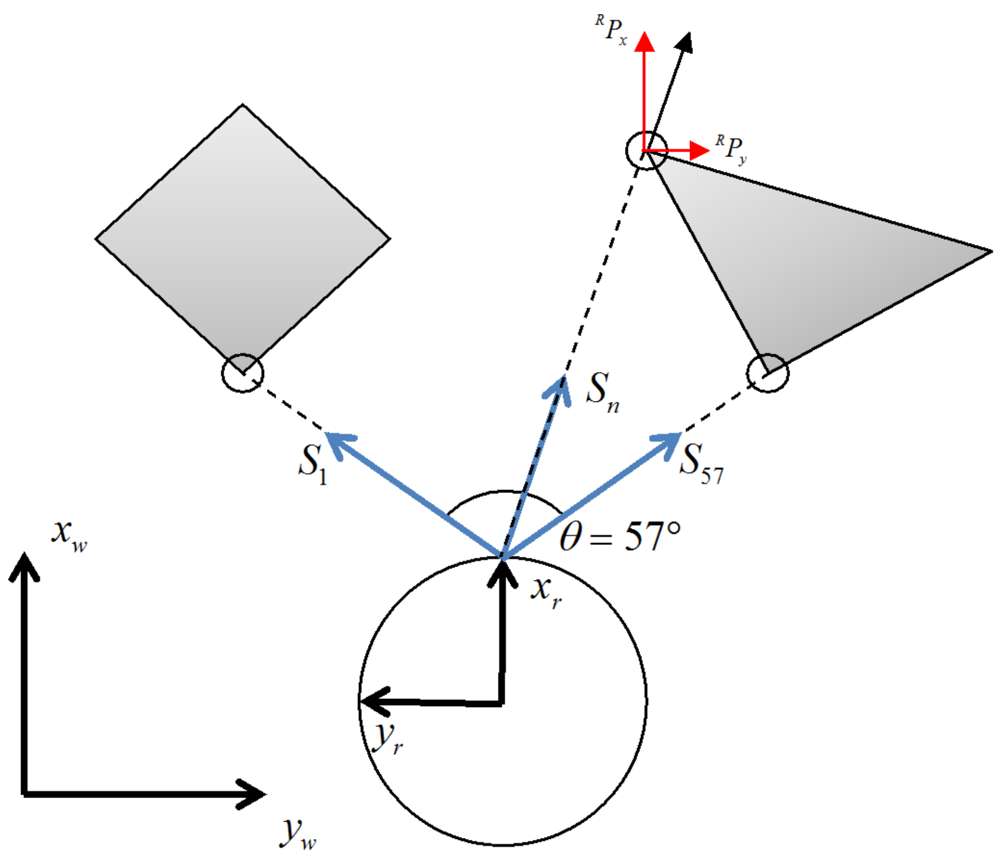


Figure 13. Diagram of the combined world and robot coordinate systems. Each represents an individual angle in the total field of view and can be treated as if it were an individual sonar sensor.

Each data point is then transformed using Equation (4) and stored in an array as two separate columns as the new *x* and *y* coordinates in the robot frame of reference. The algorithm was first tested in a long hallway to ensure accurate depth measurements for a parallel wall and plotted in polar coordinates. The output of the two-dimensional transformation is shown in Figure 14.

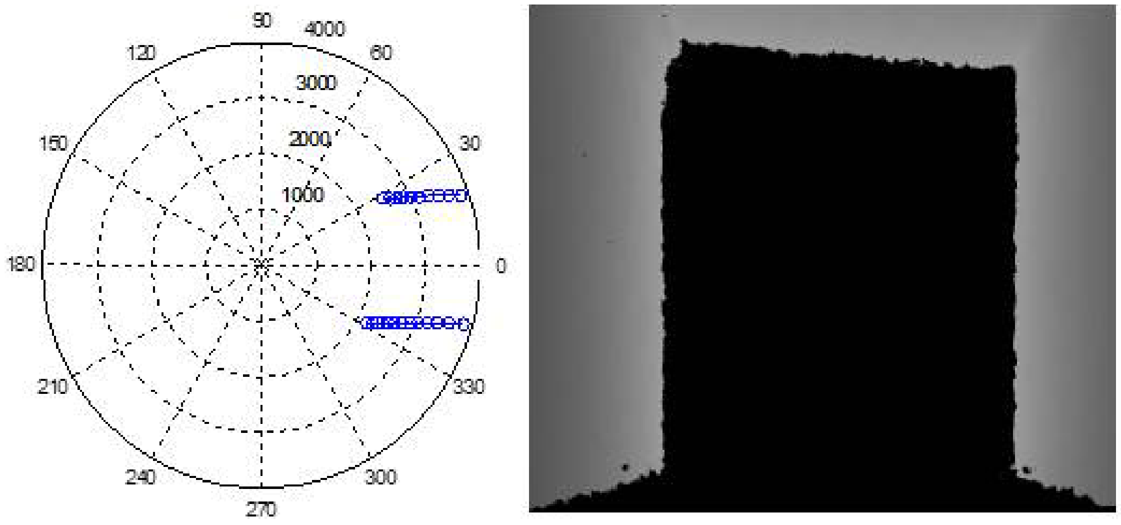


Figure 14. Plot of long hallway in robot coordinate system using the two-dimensional transformation. The image on the left displays depths in polar coordinates, while the image on the right is the raw image before processing.

As shown in Figure 14, the infrared returns collected by the Kinect and transformed by the 2-D mapping algorithm appear to display two parallel lines, providing a visual representation of the hallway with more points clustered near to the robot and along with returns spaced out at further distances. The field of view shown gives returns at the appropriate depth distances and omits any returns from the ceiling due to the image cropping that occurs as described in Section A.

THIS PAGE INTENTIONALLY LEFT BLANK

V. OBSTACLE AVOIDANCE ALGORITHM

The classical robot movement problem is realized when a robot has to decide amongst several paths in an environment with rigid obstacles. As shown in Chapters III and IV, the Xbox Kinect is capable of recognizing obstacles within a five meter distance; however, it still must be able to choose a path around these obstacles. High complexity algorithms, such as those based on critical curves [17] are considered exact algorithms and are often used to find a solution or prove that none exist. These algorithms have long processing times and are often unnecessary when quick decisions must be made. The potential field algorithm is a heuristic algorithm that simplifies the shapes of the objects and restricts the robot motion to smaller sets [17]. The potential field-based motion planning that is implemented in MATLAB is described in this chapter.

A. FOLLOWING A NARROW CORRIDOR

The first design implemented was an algorithm designed to test the robot's ability to follow a long narrow corridor and maintain a track along a centerline. The code is shown in Appendix A first collects the Kinect sensor information and analyzes it using the 2-D coordinate processing method discussed in Chapter IV. Once a coordinate has been assigned to each degree in the field-of-view, summations are taken for all *x* coordinates and *y* coordinates as shown, respectively, in

$$X_{sum} = \sum {}^{R}P_{x} \tag{6}$$

and

$$Y_{sum} = \sum {}^{R}P_{y}. \tag{7}$$

These sums calculated in Equations (6) and (7) are used to calculate the turning velocity v_{turn} and forward velocity $v_{forward}$ given, respectively, by

$$v_{forward} = k_1(X_{sum} - c) \tag{8}$$

and

$$v_{turn} = k_2(\Upsilon_{sum}). \tag{9}$$

Two constants, k_1 and k_2, are introduced into Equations (8) and (9), respectively, in order to adjust the gain of the forward velocity and turn rate. The final values of k_1 and k_2 are determined while testing the algorithm in the laboratory. It was found that the program executed best when k_1 fell between 4 and 5 and when k_2 was set to 3.5. Additionally, an offset c was used in order to adjust for coming into direct contact with a wall. It was found that the program executed best when the offset was set to zero for the hallway and set to one for the computer laboratory environment.

B. POTENTIAL FIELDS

The potential fields algorithm uses the idea of attractive and repulsive forces in order to provide a method of local path planning to avoid obstacles. The algorithm is not complete in that it can move the robot away from obstacles toward a solution but cannot calculate if a solution does not exist. To illustrate the function of the potential field algorithm one may further consider similar potential fields existing in nature such as when a positively charged particle moves towards negatively charged particle and is likewise repelled from another positively charged particle. A diagram illustrating how a robot using the potential fields algorithm is expected to move around obstacles from $\boldsymbol{q}_{start}$ to $\boldsymbol{q}_{end}$ is shown in Figure 15. The movement shown in the diagram demonstrates how the algorithm stores and releases potential energy when moving the robot from a higher to lower energy configuration.

In order to adapt the potential field algorithm to work with the Xbox Kinect and the P3-DX robotic platform, we first define the two-dimensional field using the robotic coordinate system $x_r - y_r$ shown in Figure 5 and the world coordinate system $x_w - y_w$ discussed in Chapter IV. Recall that the Xbox Kinect directs its infrared beam in the positive $\boldsymbol{x}_r$ direction, which is also the forward direction of motion of the robot. From the world coordinate system, the robot's position is denoted $\boldsymbol{q} = \begin{bmatrix} \boldsymbol{x} & \boldsymbol{y} \end{bmatrix}^T$, and the orientation of the robot θ_r is measured as the angular distance between $\boldsymbol{x_r}$ and $\boldsymbol{x_w}$.

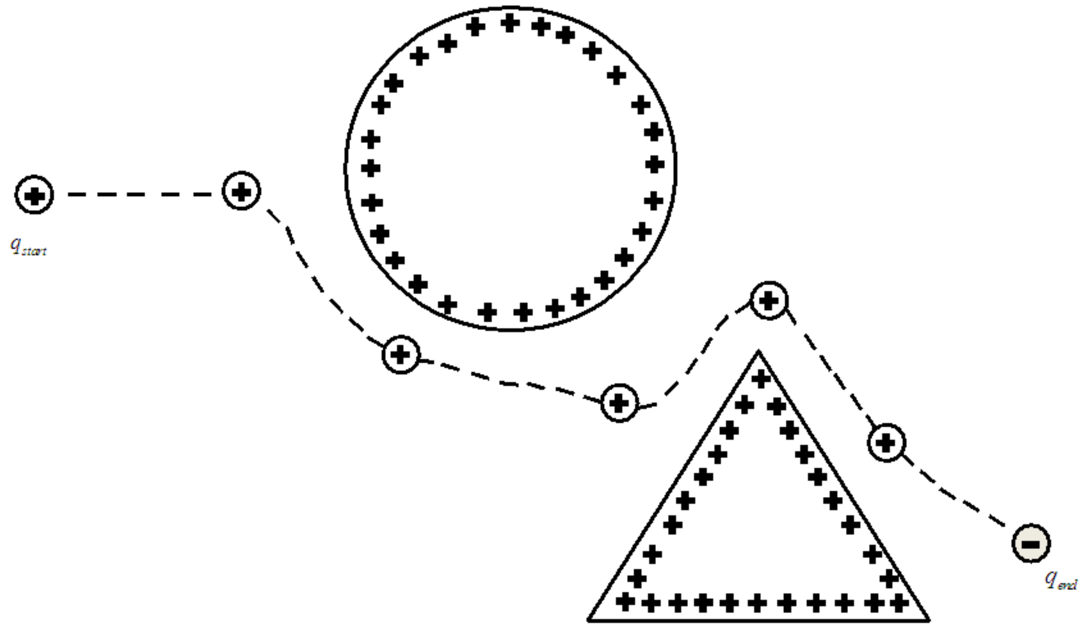


Figure 15. Illustration of desired robotic movement through a potential field from starting point q_{start} to ending point q_{end}.

The attractive potential equation used to guide the robot towards its goal is given by [18]

$$U_{att} = \begin{cases} \frac{1}{2}\xi \| q - q_{goal} \|^2, & if \| q - q_{goal} \| \le \rho \\ \xi\rho \| q - q_{goal} \|, & if \| q - q_{goal} \| > \rho \end{cases} \tag{10}$$

where q_{goal} is the goal position, ρ is a constant distance, and ξ is a constant coefficient. The attractive force can be defined using the negative gradient of the attractive potential and is given by [18]

$$^{w}F_{att} = -\frac{\partial U_{att}}{\partial q} \tag{11}$$

and

$$F_{att} = \begin{cases} -\xi(q - q_{goal}), & if \; \| q - q_{goal} \| \leq \rho \\ -\xi\rho \dfrac{(q - q_{goal})}{\| q - q_{goal} \|}, & if \; \| q - q_{goal} \| > \rho \end{cases} . \tag{12}$$

On the other hand, the repulsive forces are generated by the presence of obstacles, and each depth measurement made by the Kinect provides a repulsive force defined by [18]

$$U_{rep,i} = \begin{cases} \dfrac{1}{2}\eta\left(\dfrac{1}{d_i} - \dfrac{1}{d_c}\right)^2, & if \; d_i \leq d_c \\ 0, & if \; \mathrm{d}_i > d_c \end{cases} \tag{13}$$

where d_i is the measurement of S_i, the sensor value of the robot, d_c is a constant cutoff distance and η is a constant coefficient. The corresponding repulsive force given by [18]

$$^{r}F_{rep,i} = \left\{ \begin{matrix} -\eta\left(\dfrac{1}{d_i} - \dfrac{1}{d_c}\right)\left(\dfrac{n_i}{d_i}\right), & if \; d_i \leq d_c \\ 0, & if \; d_i > d_c \end{matrix} \right\} \tag{14}$$

where the superscript *r* indicates that the repulsive force is represented in the robot coordinate system, n_i is a unit vector in the beam direction of S_i, the sensor value of the robot. In the robot-fixed coordinate system, n_i is given by

$$n_i = \begin{bmatrix} \cos(\theta_i) \\ \sin(\theta_i) \end{bmatrix} . \tag{15}$$

The potential field algorithm was implemented using a rectangular wooden plank and a cylindrical rod in an open laboratory environment. The two obstacles were chosen due to characteristic of being narrow enough to avoid detection by the on-board sonar system. The potential field algorithm was implemented in MATLAB and is shown in Appendix B.

VI. RESULTS

A. TWO-DIMENSIONAL MAPPING

Several objects were mapped by both the P3-DX sonar ranging system and the Xbox Kinect in the two-dimensional plane. The objects were placed between 2.0 and 3.0 m in depth in front of the robot. Subsequently, a flat cardboard background was placed 1.5 m behind each object to provide a depth comparison. The series of objects were used to provide a sampling of both common and irregular shaped objects found in indoor environments and test the performance of the two-dimensional mapping algorithm for the Xbox Kinect. In each of the following subsections, the cropped depth image captured by the IR sensor is shown followed by the corresponding maps of the returns from the sonar array and the Kinect Depth Sensor.

1. Flat board

The cropped depth image captured by the Kinect of a flat cardboard surface at an approximate depth of 2.0 m is shown in Figure 16. The flat board can be identified as the gray mass in the center of Figure 16, while the lighter gray object on the left side of the gray mass is the cardboard background.

The two-dimensional map of the image shown in Figure 16 is shown in Figure 17. Two clusters of data points (in blue) can be identified on this map, while robot is located on the origin (in green). The center of the flat cardboard can be identified at (2, 0), while the center of the background mass can be identified at (1, 3.5).

The two-dimensional map of the sonar returns for the flat cardboard object is shown in Figure 18. Only two data points are returned by the sonar. The first data point, located near (2.0, 0.5), is the return from sonar 4 and gives the distance to the flat cardboard. The second return is located near $(0.0, -1.5)$ and is the sonar return off of the sidewall to the right of the robot.

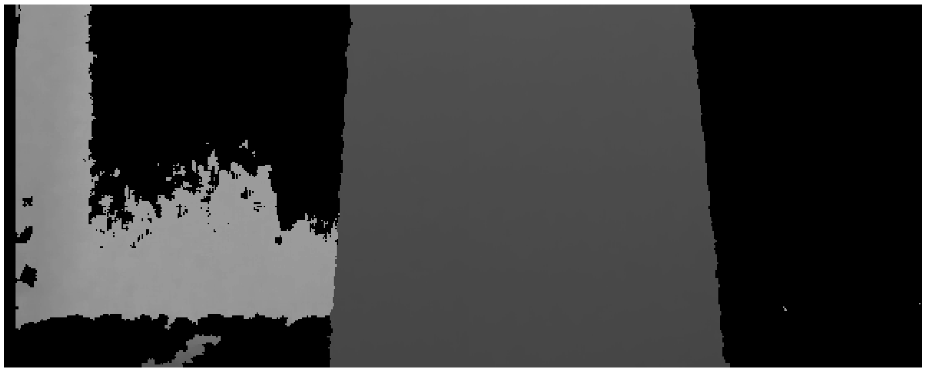


Figure 16. Cropped depth image of a flat cardboard surface.

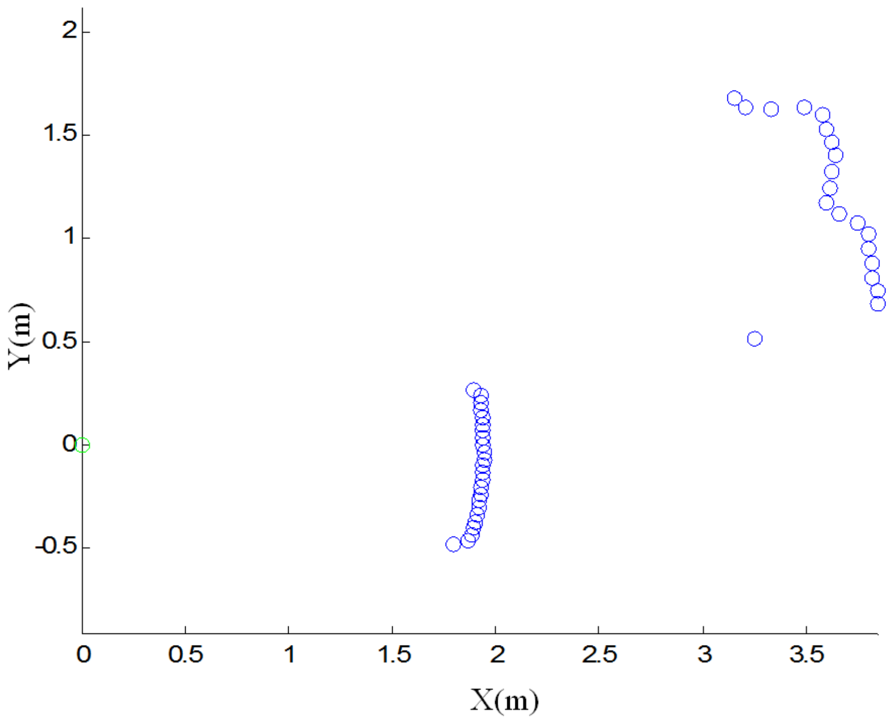


Figure 17. Two-dimensional plot of Kinect returns for a flat cardboard surface. The robot is located at the origin of the *xy*-plane and marked by a green circle.

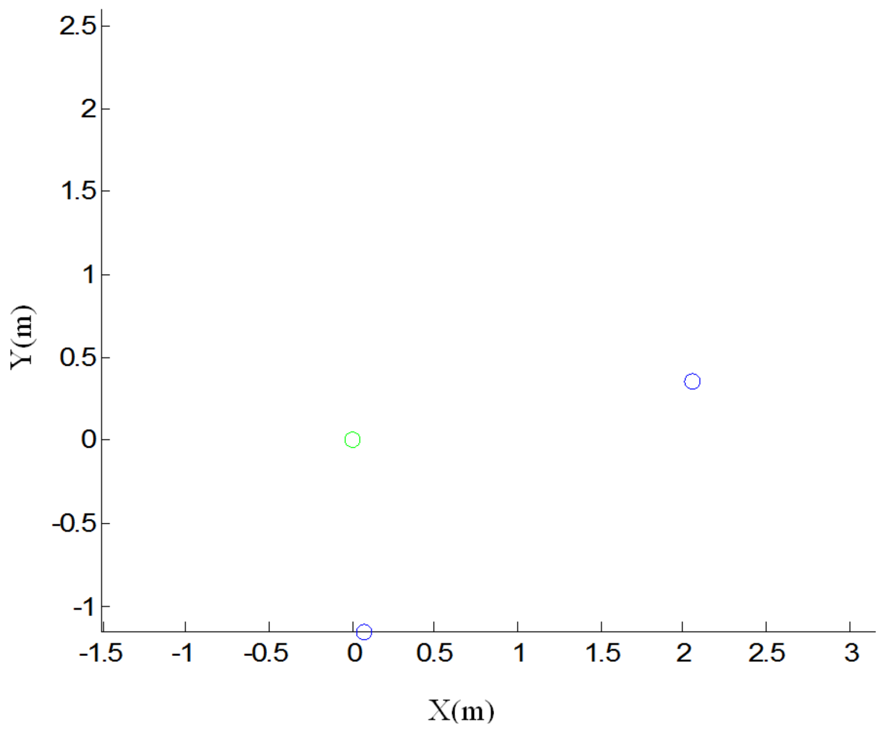


Figure 18. Two-dimensional plot of sonar returns for a flat cardboard surface. The robot is located at the origin of the *xy*-plane and marked by a green circle.

2. Desk Chair

The cropped depth image captured by the Kinect of a standard desk chair at an approximate depth of 2.0 m is shown in Figure 19. The chair is easily identified in the center of the image with the seat and back rest facing the camera. The lighter gray background is also shown to the left and the right of the chair.

The two-dimensional map of the Kinect returns is shown in Figure 20. The front of the chair can be identified at (0.1, 2.3), while the cluster of points near (0, 2.7) can be identified as the back rest. The cluster of data points behind the chair at a distance of 3.7 m from the origin can be identified as the cardboard background.

Lastly, the two-dimensional map of the sonar returns is shown in Figure 21. The sonar returns two data points from sonar 4 and sonar 8, identifying the front of the chair at (2.2, 0.3) and the adjacent wall at $(0, -1.5)$.

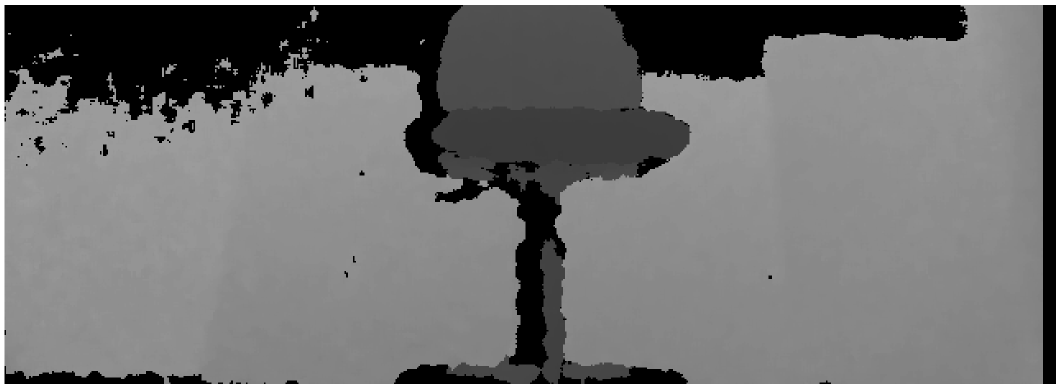


Figure 19. Cropped depth image of a desk chair.

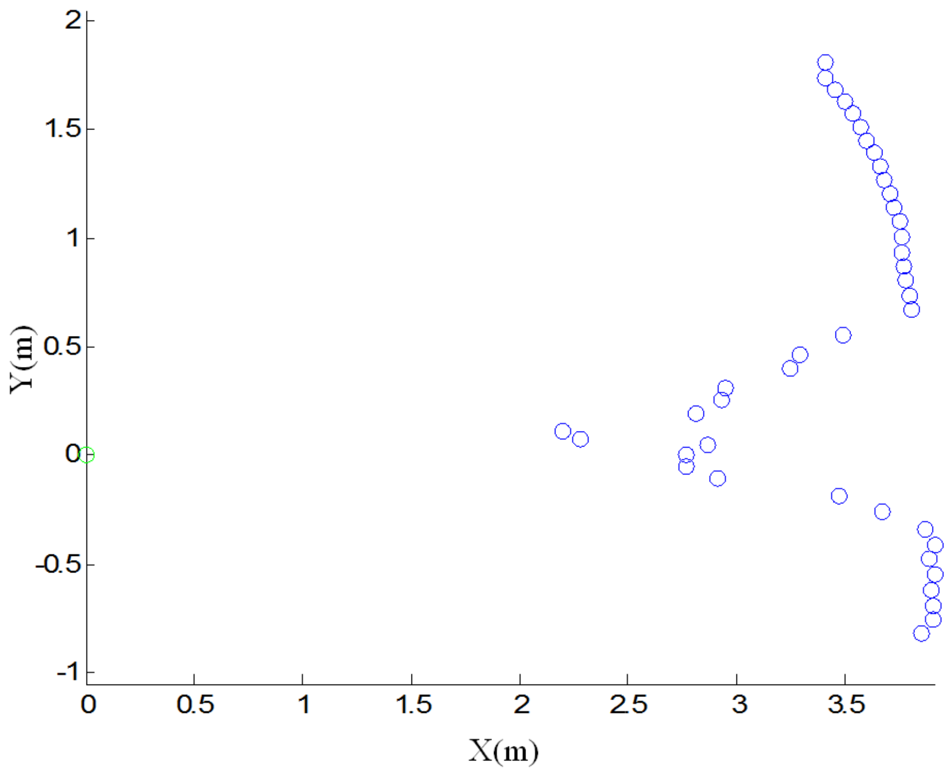


Figure 20. Two-dimensional plot of Kinect returns for a desk chair. The robot is located at the origin of the *xy*-plane and marked by a green circle.

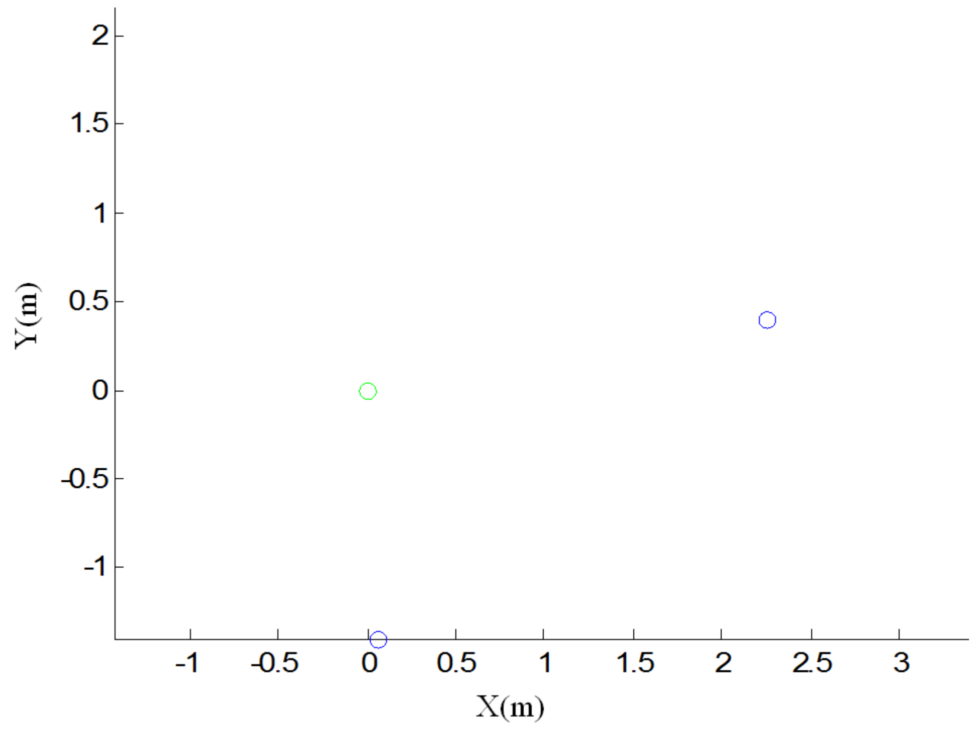


Figure 21. Two-dimensional plot of sonar returns for a desk chair. The robot is located at the origin of the *xy*-plane and marked by a green circle.

3. Desk

The depth image of a small desk at an approximate depth of 2.0 m is shown in Figure 22. The desk is located at the center of the image, and its four legs can easily be distinguished. The lighter grey cardboard background is shown to the left of the desk. The two-dimensional map of the Kinect returns for the desk is shown in Figure 23.

The square cluster centered about (2.0, 0.0) displays the returns for the four legs and backdrop of the desk. The second cluster centered at (3.5, 1.0) are the returns from the cardboard background.

The two-dimensional map of the sonar returns for the desk is shown in Figure 24. Once again, only two ranges are returned for the sonar locating the back corner of the desk near (3.0, 0.5). Sensor 8 returns the side wall at 1.5 m to the right of the robot.

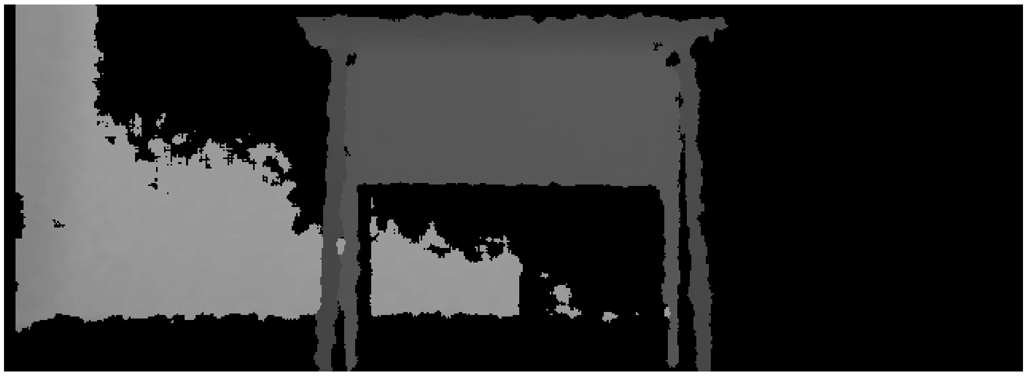


Figure 22. Cropped depth image of a four legged desk.

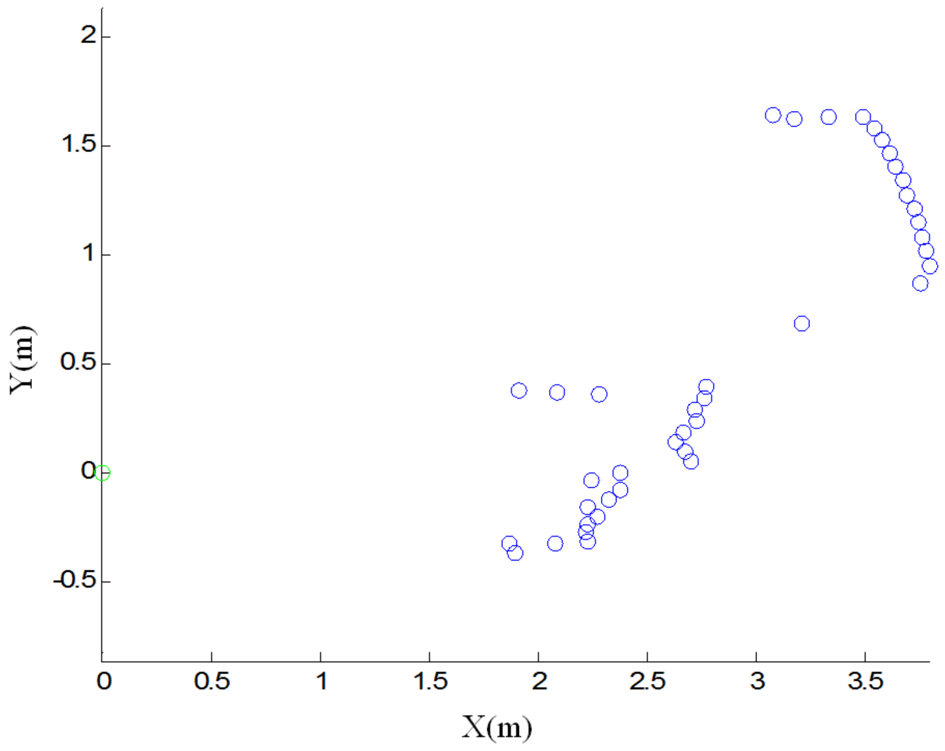


Figure 23. Two-dimensional plot of Kinect returns for a desk. The robot is located at the origin of the *xy*-plane and marked by a green circle.

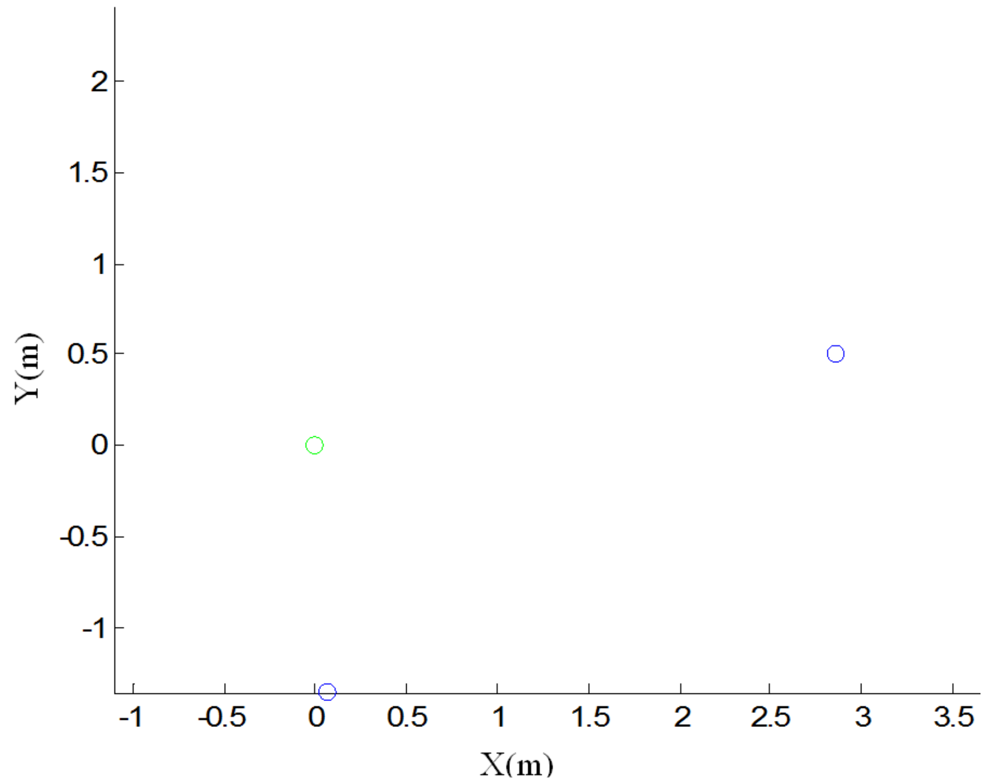


Figure 24. Two-dimensional plot of sonar returns for a desk. The robot is located at the origin of the *xy*-plane and marked by a green circle.

4. Thin Pole

The depth image of a thin pole at an approximate depth of 2.5 m is shown in Figure 25. The pole is located in the center of the figure and is separated from the lighter grey cardboard background.

The two-dimensional map of the Kinect returns for the thin pole is shown in Figure 26. Approximately two returns centered at around (0.1, 2.5) are shown in Figure 26 and represent the thin pole. The sonar returns are shown in Figure 27. No returns were recorded by the forward facing sonars for the thin pole.

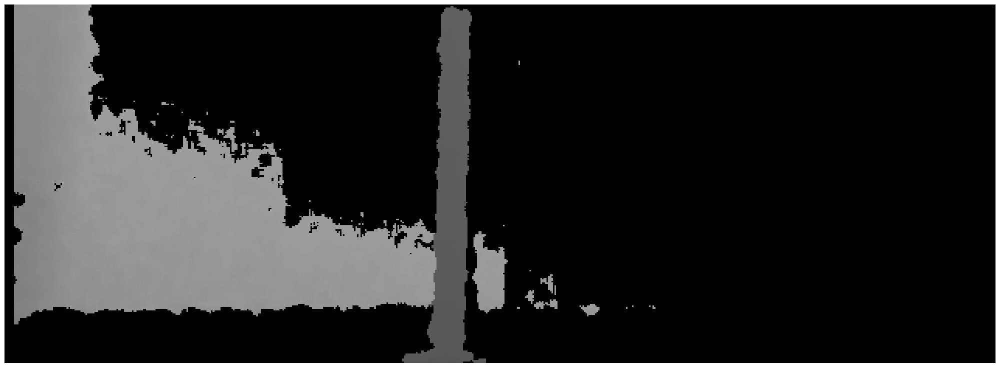


Figure 25. Cropped depth image of a thin pole.

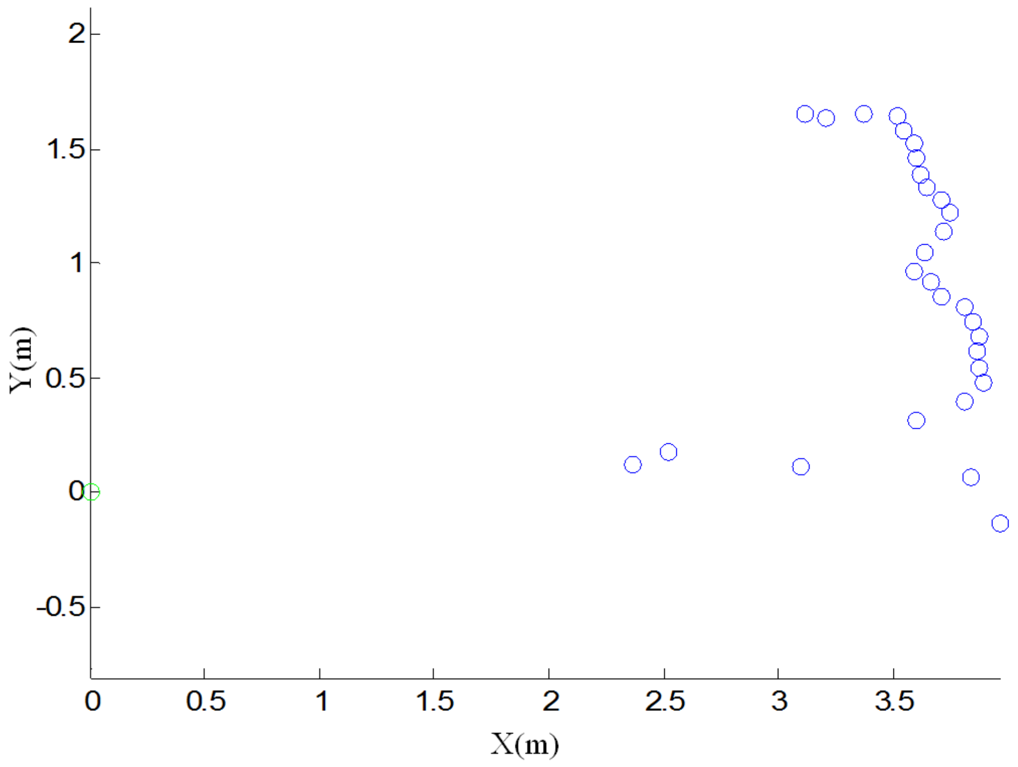


Figure 26. Two-dimensional plot of Kinect returns for a thin pole. The robot is located at the origin of the *xy*-plane and marked by a green circle.

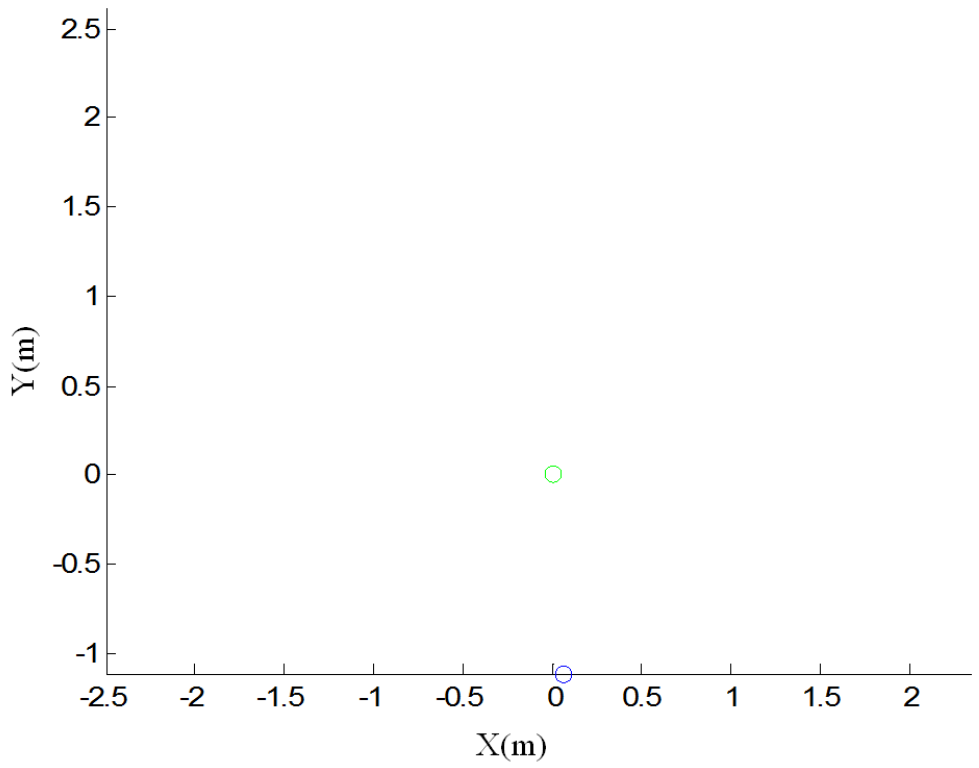


Figure 27. Two-dimensional plot of Kinect returns for a thin pole. The robot is located at the origin of the *xy*-plane and marked by a green circle. No forward facing returns were recorded.

5. Multiple Thin Poles

The cropped depth image captured by the Kinect of two thin poles at an approximate depth of 2.2 m and 0.75 m in spacing is shown in Figure 28. The two poles are found in the center of the image. A slight parallax effect occurring directly to left of the right pole can be seen in the area where the cardboard background has zero returns. The two-dimensional map of the Kinect returns is shown in Figure 29.

The two parallel data clusters closes to the origin can be identified as the two thin poles separated from the background. The two-dimensional map of the sonar returns are shown in Figure 30. Once again, the forward facing sensors do not pick up either of the thin poles.

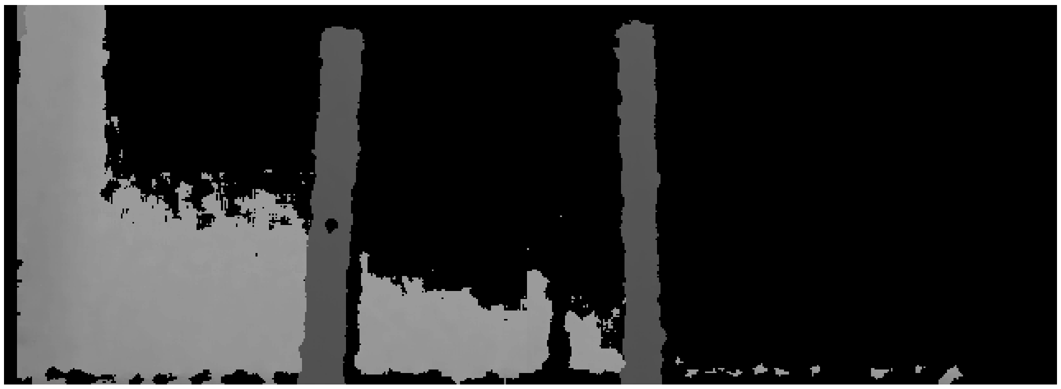


Figure 28. Cropped depth image of a two thin poles.

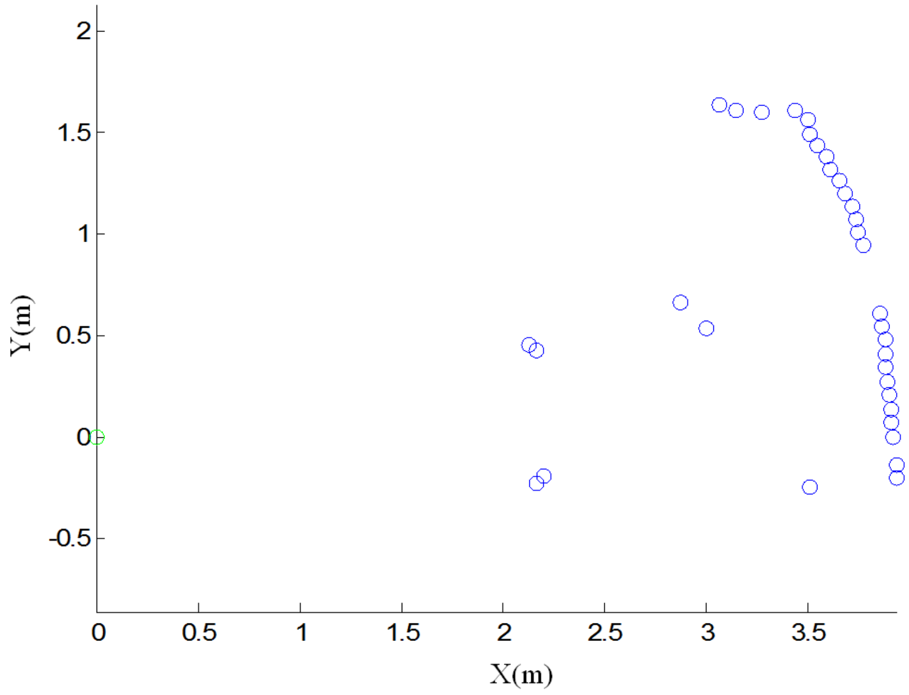


Figure 29. Two-dimensional plot of Kinect returns for two thin poles. The robot is located at the origin of the *xy*-plane and marked by a green circle.

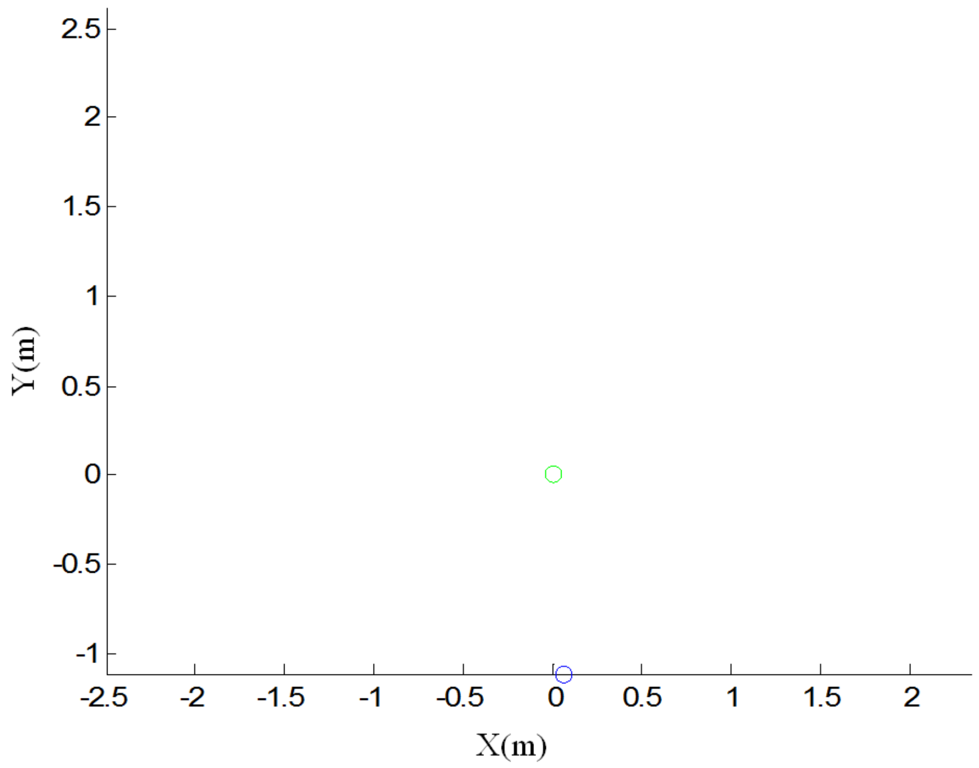


Figure 30. Two-dimensional plot of sonar returns for two thin poles. The robot is located at the origin of the *xy*-plane and marked by a green circle. No forward facing returns were recorded.

6. Test Scenario

The cropped depth image captured by the Kinect of the test scenario for the potential field algorithm is shown in Figure 31. The test scenario consists of two poles spaced 2.0 m apart and staggered nearly a meter in width in front of the robot. The background appears in lighter grey behind the first pole on the left.

The two-dimensional map of the Kinect returns is shown in Figure 32. The tightly knit cluster of three data points represents the first pole at (0.5, 1.5), while the second pole is located at (−0.5, 3.5). The dimensional map of the sonar returns are shown in Figure 33. The left pole can be identified as the data point closest to the origin; however, the right pole is not picked up by the sonar.

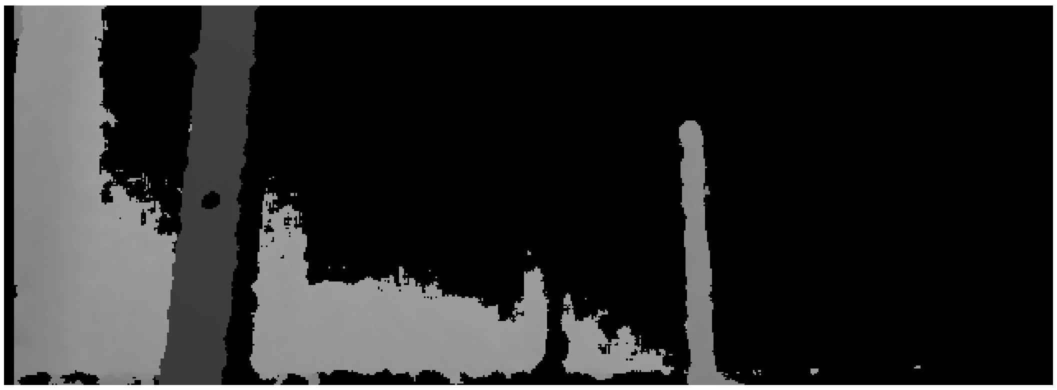


Figure 31. Cropped depth Image of the potential fields test scenario.

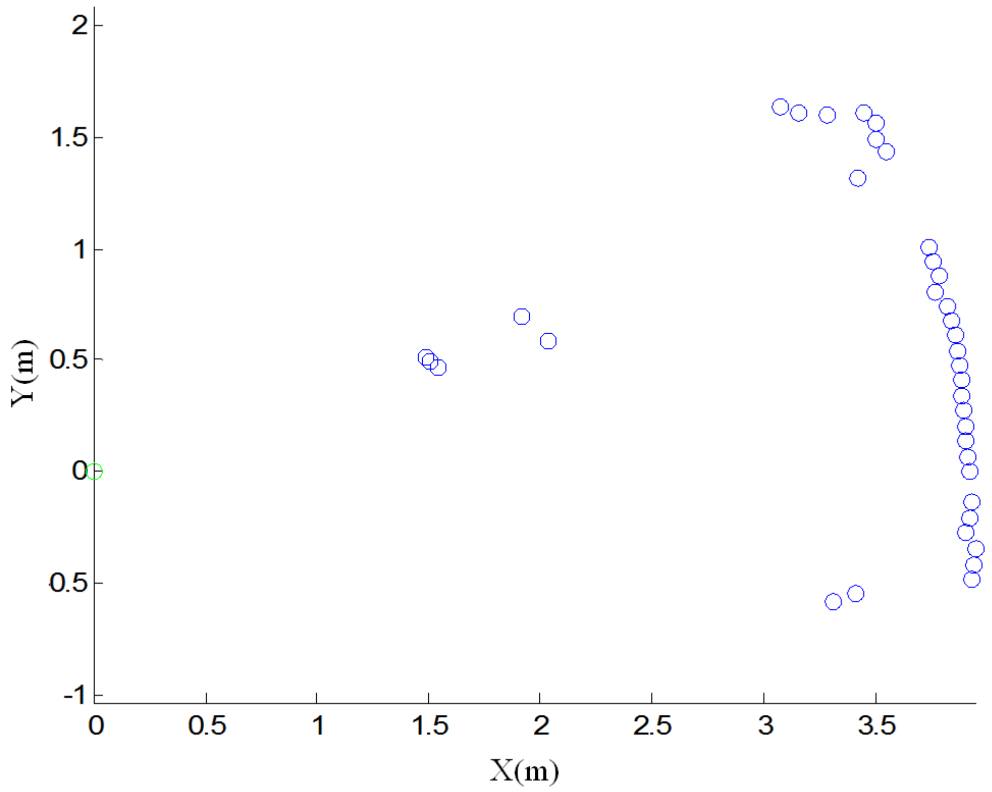


Figure 32. Two-dimensional plot of Kinect returns for the potential fields test scenario. The robot is located at the origin of the *xy*-plane and marked by a green circle.

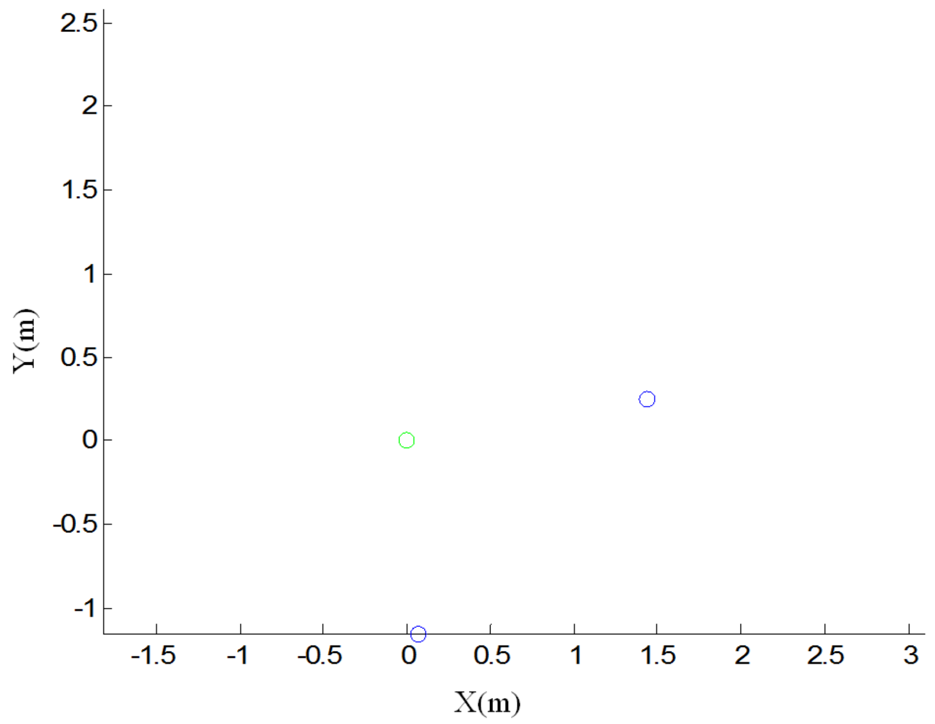


Figure 33. Two-dimensional plot of sonar returns for the potential fields test scenario. The robot is located at the origin of the *xy*-plane and marked by a green circle.

B. OBSTACLE AVOIDANCE ALGORITHMS USING P3-DX

Both obstacle avoidance algorithms were implemented without the use of the onboard sonar system and tuned to provide nearly optimal results for the indoor computer laboratory environment. The test scenario, depicted in Section A, part 6 of this chapter was used to test the potential field algorithm, while the narrow hallway shown in Figure 14 was used to test the robot's capability to follow a centerline in a hallway.

1. Narrow Corridor Following Algorithm

The P3-DX used the algorithm discussed in Chapter IV to follow the narrow corridor depicted in Figure 14. The robot was set up to traverse 5.0 m of a narrow corridor with starting points in the middle of the corridor, to the left side of the of the centerline and to the right side of the centerline. In all three cases, the algorithm shifted the P3-DX towards the centerline and the 5.0 m distance was traversed. The code used as well as the constant values used to provide nearly optimal control over the P3-DX is shown in Appendix A.

2. Potential Fields Algorithm Implementation

A demonstration of the P3-DX platform using the potential fields algorithm to maintain a centerline from q_{start} to q_{goal} through narrow obstacles is shown in Figure 14. The initial position for the robot was at the centerline of the classroom, and its final position was 4.0 m in the robot's *x*-direction. The robot was observed to be moving towards the initial wooden obstacle and avoided the obstacle by shifting its position to the right and down the centerline of the room. It then took a second turn before the second obstacle and avoided the obstacle by shifting to the left. Once the pass was complete, the robot continued moving down the centerline of the room until it reached q_{goal}. The code controlling this implementation is shown in Appendix B.

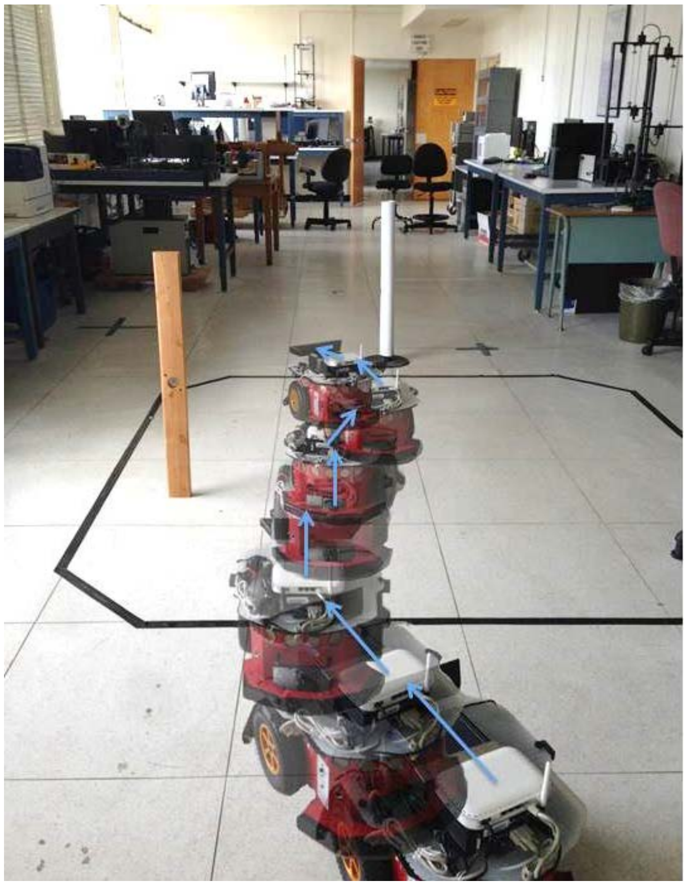


Figure 34. Robot using potential fields algorithm to avoid two narrow obstacles.

VII. SUMMARY AND CONCLUSION

Obstacle detection has been a major consideration for mobile robotics since its inception. One area of particular interest in modern research is small, lightweight systems that can be used in urban environments or onboard ships where the terrain is often unpredictable and the ability to use GPS is limited. Previous research in obstacle avoidance has focused on using systems that use large cameras such as computer vision systems or laser rangefinders that can cost up to several orders of magnitude more than the low cost autonomous system that it would be deployed on.

The Kinect utilizes an IR sensor housed inside a lightweight plastic case with the capability to provide accurate depth information of indoor environments at a relatively low cost and low power. An algorithm to analyze depth data from the Xbox Kinect sensor and transform it into a two-dimensional map of the surrounding environment was presented in this thesis. The data was then used to provide a method of obstacle detection and avoidance on the P3-DX mobile robotic platform which had previously used a sonar array to perceive its environment.

The specific characteristics and limitations of the Kinect were compared with the sonar array onboard the P3-DX platform, and an algorithm was produced that segmented the three-dimensional depth images from the Kinect into 57 different ranges, each representing one angle in the sensors field-of-view. These ranges were transformed from the sensors coordinate system to the robot coordinate system in order to provide a higher resolution map in the field-of-view of the sensor.

Several objects were mapped into the two-dimensional plane and compared with the returns provided by the sonar array previously used by the P3-DX robot. The results plotted and shown in Figures 16 to 33 are encouraging because the resolution provided by the Kinect is much greater than what the sonar system was able to provide.

Obstacle avoidance scenarios were then performed using two separate algorithms, a narrow corridor following algorithm and a potential field algorithm. The results demonstrated that in a structured testing environment, the Xbox Kinect can be used to detect and avoid narrow obstacles that are not immediately recognized by the onboard sonar array. In this way, the objective of this thesis was realized.

A. CONCLUSION

The capability of the Kinect to obtain a perception of a three-dimensional indoor environment was invaluable. Testing revealed that the mapping capabilities of the Kinect far exceed those of the P3-DX sonar array in a structured testing environment and also provided many areas of improvement that need to be considered in future work. The Kinect suffered from limited range accuracy and lost objects closer than 50.0 cm during testing. This was known prior to work with the Kinect as demonstrated by the range plot in Figure 11, and this made the control implementation of the Potential Field Algorithm much more difficult to work with than with the previously used sensor array. In initial phases of testing, the P3-DX suffered from a wobble caused by a large repulsive force resulting from several objects it was tracking in the background, while smaller objects that appeared in the foreground were lost.

An attempted solution to this wobbling problem was the use of a weighted recursive filter implemented in MATLAB that allowed past returns to be stored and provided a smoothing effect to the wobbling that occurred when multiple narrow data points were observed. This recursive filter did not work using the same weights in every scenario tested and added more data processing time to the algorithm. The final system did not use a recursive filter in its obstacle avoidance algorithm. The narrow corridor following algorithm was implemented in the same test scenario as the potential fields algorithm and in most cases worked better than the potential field implementation.

The Kinect suffered from a slight amount of lens distortion that caused objects to appear curved when they are mapped at close ranges. At longer ranges, this lens distortion disappears, but this may become an issue if the Kinect is to be used on platforms smaller than the P3-DX when avoiding obstacles at closer ranges and less spacing.

Testing the Kinect with the P3-DX platform also revealed that the projection pattern is often absorbed by darker objects. One example of this can be seen when comparing Figures 9 and 10. In Figure 9, a black cylinder appears in front of the desk on the right side of the image. In Figure 10, which displays the depth image, the cylinder does not appear in front of the desk and is instead invisible from the IR sensor.

Lastly, the MATLAB software on the processing unit used to run the obstacle avoidance algorithms often crashed due to the amount of data being pulled in by the Xbox Kinect and the P3-DX. After multiple attempts at debugging the system failed, the goal became to optimize the MATLAB code in order to save only the information that was necessary for basic processing. This meant that the features that made the MATLAB code user friendly, such as plotting a world map with the robot position and surrounding returns, were no longer used; thus, the algorithm implemented in MATLAB should be transposed to C in order to function on another program used to communicate with the P3-DX platform.

B. AREAS FOR FUTURE WORK

Due to the time constraints provided by developing, examining and implementing this thesis in six months, not all areas that we intended to cover at the beginning of this exploration were realized. Some further areas of work that may be carried out in the future are discussed in this section.

The capabilities of the Kinect were not optimized in this thesis since height of the objects was not taken into consideration. In future work a three-dimensional occupancy map should be designed rather than the two-dimensional slice this thesis provided. One way in which this might be realized is through the use of the servo motor on board the Xbox Kinect, which can operate between -27 and 27 degrees in a loop and could provide the height of the object relative to the height of the robot. This would allow the robot to travel under obstacles such as tables rather than avoiding them completely.

Additionally, a more complex path planner should be designed to both map the surrounding environment and use an A star search algorithm to find a path around obstacles in an uncertain environment. Combining multiple sensors with the Xbox Kinect would be ideal in ensuring that a full 360 view was captured rather than just the front facing line-of-sight. Furthermore, the implementation of any further point cloud calculations should be transposed from MATLAB to C or C++ to allow better processing power and more efficient timing.

The results of this research suggest that the hardware used to create the IR depth imaging can be removed from the plastic case it is housed in in order to be used on smaller autonomous systems than the P3-DX platform. The Kinect is relatively inexpensive, and the algorithms presented in this research should be expanded to make them more robust in a multitude of indoor environments.

APPENDIX A. NARROW CORRIDOR FOLLOWING CODE

```
% Author: Taylor K. Calibo
% Last Update: 27MAY2014
% Adapted from Template program for EC4310 lab 4.
% This program uses the Xbox Kinect to guide the P3-DX Mobile Platform
to
% maintain a centerline in a narrow corridor and avoid obstacles in its
% path.
% Co-author: Capt. Joshua Lum

close all;
clear all;



% define minimum clearance for stopping the robot.
MIN_CLEARANCE = 250;
sonar_clearance= 500;
rangeClearance = true;  % for use in while-loop
sonar_check=0;






% CONNECT TO THE ROBOT
p3_start
pause(5)
    %Then connect to the camera
    vid = videoinput('kinect', 2, 'Depth_640x480');
    %vidrgb=videoinput('kinect',1);

    src = getselectedsource(vid);
    %srcrgb=getselectedsource(vidrgb); %color video

    vid.FramesPerTrigger = 1;
    %vidrgb.FramesPerTrigger=1;

    src.CameraElevationAngle = 14;
    %srcrgb.CameraElevationAngle= 15;
    triggerconfig(vid, 'manual');
    start(vid);%takes half a second to start
    pause(1);
preview(vid);



% get the basic robot parameters
robotParams = p3_getRobotInfo;
numSonars = robotParams(4);
numFrontBumpers = robotParams(5);
numRearBumpers = robotParams(6);

while(p3_bumpersClear(numFrontBumpers, numRearBumpers) &&
rangeClearance)
```

```
    %Checks what sonar ranges are;
    sonarRanges = p3_getAllSonarRange(numSonars);
    sonar_check=~isempty(find(sonarRanges<sonar_clearance));
    sonar_check=0; %SET SONAR CHECK =0, ONLY KINECT WILL BE USED.

    %SECOND PART OF CODE IS USED TO TEST WITH A SONAR IF KINECT RANGE IS
    %OUT OF BOUNDS

 if sonar_check==1; %check to see if less than min distance
      % get the sonar readings
      disp('depthmode: sonar')


                                 for i = 1:8
                                   x_sn(i) = sonar_location(i,1);
                                   y_sn(i) = sonar_location(i,2);
                                   alpha(i)= sonar_location(i,3)*pi/180;

             % object position in the sonar i coordinate in meters
                                   p_sn = [sonarRanges(i)/1000 0 0 1]';

              % transform of sonar i frame relative to the robot frame
                     T_r_sn = [cos(alpha(i)) -sin(alpha(i)) 0 x_sn(i);
                               sin(alpha(i))  cos(alpha(i)) 0 y_sn(i);
                                            0 0 1 0;
                                            0 0 0 1];
                                   % object position in the robot frame
                                   p_r(:,i) =  T_r_sn * p_sn;
                                 end

                                 % initialize the sum vector
                                 sum_x = 0.0;
                                 sum_y = 0.0;

                                 % sum up the 8 front sonar vectors
                                 for i=1:8
                                     sum_x = sum(p_r(1,:));
                                     sum_y = sum(p_r(2,:));
                                 end

                                 k1=3.5;
                                 k2=1.2;
                                 offset=0;

                                 forward_vel = k1*(sum_x-offset);
                                 turning_vel = k2*sum_y;

                                 if hypot(forward_vel,turning_vel)<3.0
                                     turning_vel=7;
                                 end

                                 % set the motion velocity
                                 p3_setTransVel(forward_vel);
                                 p3_setRotVel(turning_vel);



                                 % make sure we are not too close to an
obstacle.  If we are, then set
                                 % the "rangeClearance" to False so that
```

```
we break out of the while loop
                                index = find(sonarRanges <
MIN_CLEARANCE);

sonar_check=~isempty(find(sonarRanges<sonar_clearance));

                                if(~isempty(index))
                                    rangeClearance = false;
                                    myString = 'Not enough clearance.
      Disconnecting from robot';
                                    myString2 = 'Sonar number:  ';
                                    disp(myString);
                                    disp(myString2);
                                    index-1
                                end

                                % or if we are using MobileSim and we
just want to break out of this
                                % loop, we can press the space bar (32)
                                userInput = keyinfo;
                                if(userInput(1) == 32)
                                    rangeClearance = 0;
                                    myString = 'Ending the program...';
                                    disp(myString);
                                end

                                % wait a little bit for robot to catch
up with Matlab
                                pause(1);

 else

    %get the sonar readings
    %get sensor readings there are 57 of them.
        pause(1)
        %%%Collect 1 snap of video data


        %start(vidrgb);

        imgDepth = getsnapshot(vid);
        %rgb=getdata(vidrgb);

        imgDepth=fliplr(imgDepth);
        layers={250:480};
            %im=imgDepth(layers{u},8:634);
            im=imgDepth(layers{1},:);%,171:640-171);

            %each 11.2281 pixels is 1 degree field of view


            %im(im==0)=[];

            hCol=1;
            count=0;



            %for angle_bin=1:57 %For Angles in Image
            lengthAng=57;
            for angle_bin=1:lengthAng;
                for hCol=hCol:hCol+11%for columns in image
                    imCol=im(:,hCol); %For all rows in column
                    imCol(imCol==0)=[]; %if column returns zero (non
return) input empty matrix
                    temp=mean(imCol); %Take the mean of all the depths
in the column
                    count=count+1;
                    depthPixel(count)=temp;
                    %average columns
                end
                depth(angle_bin)=mean(depthPixel);
                count=0;
            end




as=-28:28;







    %sonarRanges = p3_getAllSonarRange(numSonars);
    %SonarRanges equivalent to get depth


    for i = 1:57
      x_sn(i) = 0; %For XBOX KINECT
      y_sn(i) = 0;  %For XBOX KINECT
      alpha(i) = as(i); %LOOKING STRAIGHT AHEAD

      %object position in the sonar i coordinate in meters
      p_sn = [depth(i)/1000 0 0 1]';

      %transform of sonar i frame relative to the robot frame
      T_r_sn = [cosd(alpha(i)) -sind(alpha(i)) 0 x_sn(i);
                sind(alpha(i))  cosd(alpha(i)) 0 y_sn(i);
                0 0 1 0;
                0 0 0 1];
      %object position in the robot frame
      p_r(:,i) =  T_r_sn * p_sn; %4x8 double (8 sensors

      %scatter(p_r(1,:),p_r(2,:));
      %hold on
     % disp('Capture')
      %toc
    end





    % initialize the sum vector
    sum_x = 0.0;
    sum_y = 0.0;

    % sum up the 8 front sonar vectors
    for i=1:57
```

```
        sum_x = nansum(p_r(1,:));
        sum_y = nansum(p_r(2,:));
    end

    k1=3.4;
    k2=1.0;
    offset=0;

    forward_vel = k1*(sum_x-offset);
    turning_vel = k2*sum_y;

    if hypot(forward_vel,turning_vel)<3.0
        turning_vel=4;
    end

    % set the motion velocity
    p3_setTransVel(forward_vel);
    p3_setRotVel(turning_vel);




    % make sure we are not too close to an obstacle.  If we are, then
set
    % the "rangeClearance" to False so that we break out of the while
loop
    index = find(depth < MIN_CLEARANCE);
    sonar_check=~isempty(find(depth<sonar_clearance));
    if(~isempty(index))
        rangeClearance = false;
        myString = 'Not enough clearance. Disconnecting from robot.';
        myString2 = 'Sonar number:  ';
        disp(myString);
        disp(myString2);
        index-1
    end

    % or if we are using MobileSim and we just want to break out of this
    % loop, we can press the space bar (32)
    userInput = keyinfo;
    if(userInput(1) == 32)
        rangeClearance = 0;
        myString = 'Ending the program...';
        disp(myString);
    end

    % wait a little bit for robot to catch up with Matlab
    pause(1);
 end%if sonar check
end

% stop and disconect from the robot
p3_end
%stop and disconnect from the camera
stop(vid)
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. POTENTIAL FIELD CODE

```
% Author: Taylor K. Calibo
% Last Update: 27MAY2014
% Potential Field Work Adapted from work done by James Calusdian
% Potential Field Program With Xbox Kinect
% A script to make the robot navigate around obstacles in order to reach
% predetermined coordinates set by the user. This Algorithm uses the
Xbox
% Kinect to take in Depth Information

clear all
TRANS_VELOCITY = 500;    % translational velocity of the robot
ROT_VELOCITY = 20;     % rotational velocity of the robot
my_goal = [5000,1000]; %coordinates of goal (adjustable)
dc = 5750.0; %cut off distance
rho = 250.0; %attractive force threshold distance/min distance to goal
zeta = 1.9; %weight of attractive force (adjustable) .7
eta = 5e7; %weight of repulsive force 9e7 (adjustable)
gain_tvel = .75 ; %translational velocity gain (adjustable)1.0
gain_svel = 11.8 ; %rotational velocity gain(adjustable) 10.0

attForce = []; %
attForceD = []; %attractive force is initialized
repForce = []; %repellant force is initialized
totForce = []; %sum of all forces (repForce and attForceD)

% first connect to the robot
p3_start
pause(5)

% get the basic robot parameters
robotParams = p3_getRobotInfo;
numSonars = 57; %SET TO ANGLE IN FIELD OF VIEW
numFrontBumpers = robotParams(5);
numRearBumpers = robotParams(6);



% define minimum clearance
MIN_CLEARANCE = 275;   % 300 mm
rangeClearance = true;  % for use in while-loop
minSonarID = 1;    % this has the sonar with the min reading
minRange = 3*MIN_CLEARANCE;  % initialize to something
phi = [-28:28]*pi/180; %angle of each sonar in radians

%Intialize Computer Kinect Components
 vid = videoinput('kinect', 2, 'Depth_640x480');
 src = getselectedsource(vid);
 vid.FramesPerTrigger = 1;
 src.CameraElevationAngle = 17;
 triggerconfig(vid, 'manual');
 start(vid);%takes half a second to start
 pause(1);
 preview(vid);


while(p3_bumpersClear(numFrontBumpers, numRearBumpers) &&
rangeClearance)
```

```
      sonarRanges = p3_getAllSonarRange(numSonars);

      % print sonar ranges
      %sonarRanges'

      %Get Single Snapshot From Kinect
         imgDepth = getsnapshot(vid);
         imgDepth=fliplr(imgDepth);
         im=imgDepth(250:480,:); %Crop Image

         %Initializations for For Loop
         hCol=1;
         count=0;

      lengthAng=57;
             for angle_bin=1:lengthAng;
                 for hCol=hCol:hCol+11%for columns in image
                     imCol=im(:,hCol); %For all rows in column
                     imCol(imCol==0)=[]; %if column returns zero (non
return) input empty matrix
                     temp=mean(imCol); %Take the mean of all the depths
in the column
                     count=count+1;
                     depthPixel(count)=temp;
                     %average columns
                 end
                 depth(angle_bin)=mean(depthPixel);
                 count=0;
             end



         %Store Depths
         as=-28:28; %Angle Information Total of 57 degrees

     % look at the forward sonars (0-7) to see where we have minimum
     % clearance
     minRange = 10*MIN_CLEARANCE;  % reset to something large each time
     for ii = 0:56 %57 sensors
         if(depth(ii+1) < minRange)
             minSonarID = ii;
             minRange = sonarRanges(ii+1);
         end
     end

     minSonarID;
     minRange;


     % determine robot movement
     my_Data = p3_getXYHeading; %assigns start out position of robot(x,y)
to my_Data variable

     attForce(1) = my_Data(1) - my_goal(1); %calculates distance from
robot position to robot goal in the x direction
     attForce(2) = my_Data(2) - my_goal(2); %calculates distance from
robot position to robot goal in the y direction
     attForceMag = sqrt((attForce(1))^2 + (attForce(2))^2); %calculates
magnitude of distance from robot to goal
```

```
    theta = my_Data(3) * (pi/180); %robot steering angle in robot
coordinates

    % see equation 2 in "Yun '97"
    if(attForceMag <= rho)
        attForce(1) = -1 * zeta *attForce(1); % World coordinate system
        attForce(2) = -1 * zeta *attForce(2); % World coordinate system
    else
        attForce(1) = -1 * zeta * rho * attForce(1) / attForceMag;
        attForce(2) = -1 * zeta * rho * attForce(2) / attForceMag;
    end

    attForceD(1) = attForce(1) * cos(theta) + attForce(2) * sin(theta);
% Robot coordinate system
    attForceD(2) = -1 * attForce(1) * sin(theta) + attForce(2) *
cos(theta); % Robot coordinate system

    attForceD


    repForce(1) = 0.0; %initialize repForce x-component to 0
    repForce(2) = 0.0; %initialize repForce y-component to 0

    % gathers information from sonars and compares to cut off distance
(dc)
    % in order to calculate repForce
    for ix =1:numSonars %57
        if(depth(ix) <= dc)
            temp = -1 * eta * (1/depth(ix)- 1/dc) * 1/depth(ix);
        else
            temp = 0.0;
        end
        repForce(1) = repForce(1) + temp*cos(phi(ix));
        repForce(2) = repForce(2) + temp*sin(phi(ix));
    end

    repForceMag = sqrt((repForce(1))^2 + (repForce(2))^2); % repellant
force magnitude is calcualted
    repForce

    totForce(1) = attForceD(1) + repForce(1); %.65 Total force is
calculated by summing attractive and repellant (x-component)
    totForce(2) = attForceD(2) + repForce(2); %.65 Total force is
calculated by summing attractive and repellant (y-component)
    totForce

    tempA = atan2(totForce(2), totForce(1)) * (180/pi)

    p3_setTransVel(gain_tvel * totForce(1)); % Translational velocity is
set from x-component of totForce
    p3_setRotVel(gain_svel * atan2(totForce(2), totForce(1)));
%Rotational velocity is set


    % used to stop robot once it has reached the predetermined goal
    if(attForceMag <= 350)
        pause(2);
        string = 'Destination Reached, Ending Potential Field Program';
        disp(string);
        p3_setTransVel(0);
        p3_setRotVel(0);
        rangeClearance = 0;
    end


    % Used to manually stop robot. If we want to break out of this
    % loop, we can press the space bar (32)
    userInput = keyinfo;
    if(userInput(1) == 32)
        rangeClearance = 0;
        myString = 'End Program requested, Ending potential field
program...';
        disp(myString);
        stop(vid)
    end

    % wait a little bit for robot to catch up with Matlab
    pause(1);

end

% stop and disconect from the robot
p3_end
stop(vid)
```

LIST OF REFERENCES

[1] J. Pransky, “Mobile robots: Big benefits for U.S. Military,” *Industrial Robot: An International Journal*, vol. 24, no. 2, pp.126–130, Jan. 1997.

[2] X. Lu, C. Chia-Chih and J. Aggarwal, “Human detection using depth information by Kinect,” *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops Proceedings*, 2011, pp. 15–22.

[3] P. Benavidez and M. Jamshidi, “Mobile Robot navigation and target tracking system,” *6th International Conference on System of Systems Engineering*, 2011, pp. 299–304.

[4] R. Ambrose, R. Savely, S. Goza, P. Strawser, M. Diftler, I. Spain, and N. Radford, “Mobile manipulation using NASA’s Robonaut,” *IEEE International Conference on Robotics and Automation,* New Orleans, LA, 2004, vol. 2, pp. 2104–2109, April 2004.

[5] R. Simpson, D. Poirot, F. Baxter, “The Hephaestus Smart Wheelchair System,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 10, no. 2, pp. 118–122, June 2002.

[6] T. Bui and K. Hong, “Sonar-based obstacle avoidance using region partition scheme” *Journal of Mechanical Science and Technology*, vol. 24, no. 1, pp. 365–372, Jan. 2010.

[7] I. Susnea, A. Filipescu, G. Vasiliu, and S. Filipescu, “Path following, real-time, embedded fuzzy control of a mobile platform wheeled mobile robot,” *IEEE International Conference on Automation and Logistics*, Qingdao, China, 2008, pp. 268, 272.

[8] V. Savkin and C.Wang, “A simple biologically inspired algorithm for collision-free navigation of a unicycle-like robot in dynamic environments with moving obstacles,” *Robotica*, vol. 31, no. 6, pp. 993–1001, Sept. 2013.

[9] C. Ye, “Mixed pixels removal of a laser rangefinder for mobile robot 3-D terrain mapping,” *International Conference on Information and Automation*, Hunan, China, 2008, pp. 1153–1158.

[10] N. Al-Sahib and A. Salih, “Path Planning Control for the Mobile Robot,” *Al-Khwarizmi Engineering Journal*, vol. 7, no. 4, pp. 16, Sept. 2011.

[11] B. Wei, J. Gao, K. Li, Y. Fan, X. Gao and B. Gao, “Indoor Mobile Robot Obstacle Detection Based on Linear Structured Light Vision System,” *in IEEE International Conference on Robotics and Biomimetics*, Feb. 2009, pp. 834–839.

[12] M. Viager. (2011). Analysis of Kinect for mobile robots [Online]. http://www.scribd.com/doc- 56470872/Analysis- of-Kinect-for-Mobile-Robots-unofficial-Kinect- data-sheet-on-page-27

[13] A. Oliver, S. Kang, B. Wünsche, and B. MacDonald, "Using the Kinect as a navigation sensor for mobile robotics," *In Proceedings of the 27th Conference on Image and Vision Computing New Zealand* (IVCNZ '12), ACM, New York, NY, 2012, pp. 509–514.

[14] D. Correa, D. Sciotti, M. Prado, D. Sales, D. Wolf and F. Osorio, "Mobile robots navigation in indoor environments using Kinect sensor," *2012 Second Brazilian Conference on Critical Embedded Systems*, 2012, pp. 36–41.

[15] Mobile Robotics. (2008). Pioneer 3-DX datasheet. [Online]. Available: http://www.mobilerobots.com/ResearchRobots/PioneerP3DX.aspx

[16] Mathworks. (2014). Using the Kinect for Windows from image acquisition toolbox. [Online]. Available: http://www.mathworks.com/help/imaq/examples/using-the-kinect-r-for-windows-r-from-image-acquisition-toolbox-tm.html

[17] C. Pheatt and J. Ballester. (2011). Using the Xbox Kinect Sensor for positional data acquisition. [Online]. Available: http://hdl.handle.net/123456789/173

[18] X. Yun and K. Tan, "A wall-following method for escaping local minima in potential field based motion planning," *In Proceedings of 8th International Conference on Advanced Robotics*, Monterey, CA, 1997, pp. 421–426.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California